Network Guide

# Anybus® CompactCom 40 PROFINET IO

## PROFINET RT / IRT

**Doc.Id. HMSI-27-226**
**Rev. 1.22**



HMS Industrial Networks
Mailing address: Box 4126, 300 04 Halmstad, Sweden
Visiting address: Stationsgatan 37, Halmstad, Sweden

E-mail: info@hms-networks.com
Web: www.anybus.com

# Important User Information

This document is intended to provide a good understanding of the functionality offered by the Anybus CompactCom 40 PROFINET IO. The document only describes the features that are specific to the Anybus CompactCom 40 PROFINET IO. For general information regarding the Anybus CompactCom 40, consult the Anybus CompactCom 40 design guides.

The reader of this document is expected to be familiar with high level software design, and communication systems in general. The use of advanced PROFINET IO-specific functionality may require in-depth knowledge in PROFINET IO networking internals and/or information from the official PROFINET specifications. In such cases, the people responsible for the implementation of this product should either obtain the PROFINET specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

## Liability

Every care has been taken in the preparation of this manual. Please inform HMS Industrial Networks AB of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks AB, reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be considered as a commitment by HMS Industrial Networks AB. HMS Industrial Networks AB assumes no responsibility for any errors that may appear in this document.

There are many applications of this product. Those responsible for the use of this device must ensure that all the necessary steps have been taken to verify that the applications meet all performance and safety requirements including any applicable laws, regulations, codes, and standards.

HMS Industrial Networks AB will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features, timing, or functional side effects found outside the documented scope of this product. The effects caused by any direct or indirect use of such aspects of the product are undefined, and may include e.g. compatibility issues and stability issues.

The examples and illustrations in this document are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks AB cannot assume responsibility for actual use based on these examples and illustrations.

## Intellectual Property Rights

HMS Industrial Networks AB has intellectual property rights relating to technology embodied in the product described in this document. These intellectual property rights may include patents and pending patent applications in the US and other countries.

## Trademark Acknowledgements

Anybus ® is a registered trademark of HMS Industrial Networks AB. All other trademarks are the property of their respective holders.

| | |
|---|---|
| **Warning**: | This is a class A product.In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures. |
| **ESD Note**: | This product contains ESD (Electrostatic Discharge) sensitive parts that may be damaged if ESD control procedures are not followed. Static control precautions are required when handling the product. Failure to observe this may cause damage to the product. |

Anybus CompactCom 40 PROFINET IO Network Guide
Rev 1.22

March 2015 Doc Id HMSI-27-226

# Table of Contents

## Chapter 3 PROFINET IO Implementation Details



## Chapter 4 FTP Server



## Chapter 5 Web Server



## Chapter 6 E-mail Client

## Chapter 7 Server Side Include (SSI)



## Chapter 8 JSON



## Chapter 9 SNMP Agent



## Chapter 10 Media Redundancy Protocol (MRP)

## Appendix F Technical Specification



## Appendix G Conformance Test Guide



## Appendix H Copyright Notices

# P. About This Document

For more information, documentation etc., please visit the HMS website, 'www.anybus.com'.

## P.1 Related Documents

| Document | Author |
|---|---|
| Anybus CompactCom 40 Software Design Guide | www.anybus.com |
| Anybus CompactCom M40 Hardware Design Guide | |
| Anybus CompactCom Software Driver User Guide | |
| PROFINET IO specification, rev. 2.3 | Profibus International |
| PROFIenergy Technical Specification, rev. 1.0 | Profibus International |
| GSDML Technical Specification for PROFINET IO, rev 2.31 | Profibus International |

## P.2 Document History

Summary of Recent Changes (1.21... 1.22)

| Change | Page(s) |
|---|---|
| Rewrote Configuration Mismatch section | 24 |
| Added attribute #16 (IP configuration) to the Ethernet Host Object (F9h) | 145 |
| Removed the system contact attribute of the PROFINET IO Object from the Conformance section | 159 |
| Added Documentation Considerations section | 160 |
| | |
| | |
| | |

### Revision List

| Revision | Date | Author(s) | Chapter(s) | Description |
|---|---|---|---|---|
| 0.50 | 2014-01-24 | KaD | All | First draft |
| 1.00 | 2014-06-13 | KaD | All | First public release |
| 1.10 | 2014-09-01 | KeL, KaD | All | Major updates |
| 1.11 | 2014-10-09 | KaD | All | Added Fiber Optic information, minor updates |
| 1.20 | 2014-12-15 | KaD | All | Major updates |
| 1.21 | 2015-03-06 | KaD | 3, 11, 12, D, G | Major updates |
| 1.22 | 2015-03-XX | KaD | | Minor updates |

# P.3 Conventions & Terminology

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The terms 'Anybus' or 'module' refers to the Anybus CompactCom module
- The terms 'host' or 'host application' refers to the device that hosts the Anybus module
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value
- A byte always consists of 8 bits

# P.4 Support

For general contact information and support, please refer to the contact and support pages at www.anybus.com.

# 1. About the Anybus CompactCom 40 PROFINET IO

## 1.1 General Information

The Anybus CompactCom 40 PROFINET IO communication module provides instant PROFINET Real Time connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type.

This product conforms to all aspects of the host interface defined in the Anybus CompactCom 40 Hardware- and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to take advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

## 1.2 Features

- Ethernet connectors
- Up to 128 submodules in total
- Up to 32767 ADIs
- Generic and PROFINET specific diagnostic support
- Complies with PROFINET IO Conformance class C
- Supports up to 1440 bytes I/O data in each direction, status bytes included.
- Supports 250 µs cycle time
- SNMP agent
- FTP server
- E-mail client
- Server Side Include (SSI) functionality
- JSON functionality
- Device identity customization
- GSD file template provided by HMS
- Supports PROFINET Fast Start Up
- Supports Media Redundancy Protocol (MRP)

# 1.3 Differences between the 40 and 30 series

### I&M

- I&M5-15 requests are rejected by the CompactCom 40 module in nontransparent I&M mode.
- I&M Record data transparent mode is replaced with the IM_Options command.
- I&M0 parameters IM Version and IM Supported are removed from the PROFINET Host object and set to constant values.

### Diagnostics

- Process alarms cannot be created.
- Diagnostic events can be created when not online (i.e. in PROCESS_ACTIVE or IDLE).
- The structure of network specific event information has changed. Instead of including diagnostic source information such as API, Slot and Subslot in the data field it is extracted from the extended diagnostic fields in the create command. API, Slot and Subslot is determined with the help of Slot and ADI given by the extended diagnostic mode.

### Network Configuration Object

- Network specific instances are moved from instance number 15 and on to instance number 20 and on. This is done to increase the amount of instances in the part that is not network specific.
- The network specific instances that handled I&M data have been removed, as they are not possible to set from the application.

### PROFINET Additional Diagnostic Object

- This object is removed. All diagnostics are handled via the standard diagnostic object.

### PROFINET Host Object

- The instance #1 attribute PROFIenergy functionality does no longer carry any functionality. It is now set to reserved.
- The instance #1 attribute System Contact is now set to reserved and no longer used.
- The Expected_Ident_Ind command replaces Ar_Info_Ind. See “Command Details: Expected_Ident_Ind” on page 140 for more information.

### LEDs

- Major internal error is now covered by the FATAL event. It is signaled by both Module and Status LEDs being solid red.

### SNMP MIB-II

- sysContact, sysLocation and sysName are no longer connected to any other parameters/attributes. They are stored as separable readable/writeable parameters in the module’s nonvolatile memory.

# 2. Basic Operation

## 2.1 General Information

### 2.1.1 Software Requirements

Generally, no additional network support code needs to be written to support the Anybus CompactCom 40 PROFINET IRT module, however due to the nature of the PROFINET networking system certain things must be taken into account:

- Up to 32767 ADIs can be represented on PROFINET.
- ADI names, types and similar attributes cannot be accessed via PROFINET. They are however represented on the network through the built in web server.
- Up to 5 diagnostic instances can be created by the host application. An additional 6th instance may be created in event of a major fault.
- For conformance reasons, the host application must implement support for network reset types 00h (Power-on) and 02h (Power-on + Factory Default) in the Application Object (FFh).
- PROFINET in itself does not impose any particular timing demands when it comes to acyclic requests (i.e. requests towards instances in the Application Data Object), however it is generally recommended to process and respond to such requests within a reasonable time period (exactly what this means in practice depends on the implementation and the actual installation).
- The order in which ADIs are mapped to Process Data is significant and must be replicated in the IO Controller when setting up the network communication (i.e. modules must be set up in the same order, size and direction, as the mapped ADIs). In case of a configuration mismatch, see "Configuration Mismatch" on page 24 for more information.

See also...

- "Application Data Instances (ADIs)" on page 16
- "Diagnostic Object (02h)" on page 73 (Anybus Module Object)
- Anybus CompactCom 40 Software Design Guide, 'Application Data Object (FEh)'

## 2.1.2 Electronic Data Sheet (GSD)

On PROFINET, the characteristics of a device is stored in an XML data file. This file, referred to as the 'GSD' file, is used by PROFINET engineering tools when setting up the network configuration. HMS provides an example GSD file, which must be adapted by the user to suit the application.

### Setting Identity and Function Information

The GSD file must be adapted to your implementation. First thing is the device identity.

In the GSD file there is a section called "DeviceIdentity". It looks like this.

```
<DeviceIdentity VendorID="0x010C" DeviceID="0x0010">
    <InfoText TextId="T_ID_DEV_DESCRIPTION"/>
    <VendorName Value="HMS Industrial Networks"/>
</DeviceIdentity>
```

The identity in the example represents HMS values.

- **Replace VendorID value 0x010C with the value which correspond to your vendor name.** If you do not have a Vendor ID you can obtain this by contacting PI. Set with PROFINET IO Object (0xF6) attribute #2 (Vendor ID).
- **Replace the DeviceID value 0x0010 with the value you have selected for this device.** Set with PROFINET IO Object (0xF6) attribute #1 (Device ID). Please note that if you change the Device ID you MUST also change the Vendor ID, as the Device ID is unique for the Vendor ID.
- **Replace the VendorName value "HMS Industrial Networks" with name of your vendor.** Please note that the keyword VendorName is found not only here, but also at other places in the GSD file. Use "Search" to find all instances and replace them with the name of your vendor.

Specify the function of the device.

```
<DeviceFunction>
    <Family MainFamily="General" ProductFamily="Anybus CompactCom 40 PIR"/>
</DeviceFunction>
```

The example GSD specifies a kind of "General" device as the usage of it is unclear.

- Replace the MainFamily with the class that best describe the device. The following are the allowed values:
  - "General", "Drives", "Switching Devices", "I/O", "Valves", "Controllers", "HMI", "Encoders", "NC/RC", "Gateway", "PLCs", "Ident Systems", "PA Profiles", "Network Components", "Sensors"
- Replace the ProductFamily value "Anybus CompactCom 40 PIR" with a string which describes your device.

In addition to the above, there are a few more places where identity related information is present in the GSD file.

DNS_CompatibleName="ABCC40-PIR"

For the Device access point (DeviceAccessPointItem) there is a keyword which is called DNS_CompatibleName. Locate this by using the search function.

- Replace DNS_CompatibleName value “ABCC40-PIR” with the set Station Type.
  Set with PROFINET IO Object (0xF6) attribute #3 (Station Type).

The order number of the device is set with the keyword “OrderNumber”.

<OrderNumber Value="ABCC40-PIR"/>

In many cases the value of the OrderNumber equals the Station Type string, but it does not necessary need to be that way.

- Replace the OrderNumber value “ABCC40-PIR” with the order number used for the device.
  Set with PROFINET IO Object (0xF6) attribute #8 (IM Order ID).

### How to Enable Initial Record Data

During the establishment of an IO connection between the IO device and the IO controller it is possible for the IO controller to send initial record data. This initial record data is sent using the PROFINET IO service record write. This service can be used at any time and will write data to a defined ADI. The initial record data is defined in the GSD file, and is specified for a submodule of a module. By default, the Anybus CompactCom 40 module will not make use of any initial record data, but that can be enabled if needed.

To enable this functionality, the GSD file needs to be modified as specified below. In this example, 2 bytes are written to ADI 67 (ADI 67 corresponds to index 67) during startup of a PROFINET IO connection (the value can be configured by the end user):

```
<RecordDataList>
    <ParameterRecordDataItem Index="67" Length="1" TransferSequence="0">
        <Name TextId="T_ID_EXAMPLE2"/>
        <Ref DataType="Unsigned16" ByteOffset="0" DefaultValue=”0”
            AllowedValues="0..65535" TextId="T_ID_EXAMPLE2_PRM_1"/>
    </ParameterRecordDataItem>
</RecordDataList>
```

It is recommended that the above GSD information is placed directly after the “</IOData>” keyword for the module for which the data is associated.

If more than one ADI need to be set, the keyword “ParameterRecordDataItem” is duplicated.

Please note that TextId’s (“T_ID_xxx” above) need to be added to the “<ExternalTextList>” section of the GSD file (once for each language defined).

```
<Text TextId="T_ID_EXAMPLE2" Value="Config parameter 1"/>
<Text TextId="T_ID_EXAMPLE2_PRM_1" Value="Parameter value description"/>
```

# 2.2 Network Identity

By default, the module identifies itself as a generic Anybus implementation as follows:

| | |
|---|---|
| **Device ID** | 0010h (Anybus CompactCom 40 PROFINET) |
| **Vendor ID** | 010Ch (HMS Industrial Networks) |
| **Station Type** | 'ABCC40-PIR' |

It is recommended to customize the identity information so that the Anybus module appears as a vendor specific implementation rather than a generic Anybus product.

See also...

- "Identification & Maintenance (I&M)" on page 26
- "PROFINET IO Object (F6h)" on page 130

# 2.3 Communication Settings

Network related communication settings are grouped in the Network Configuration Object (04h), and includes...

- **Ethernet Interface Settings**

  The module is locked to 100 Mbit full duplex operation as required by PROFINET.

- **TCP/IP Settings**

  These settings must be set in order for the module to be able to participate on the network. Normally set by the IO Controller.

- **SMTP Account Settings**

  These settings must be set in order for the module to be able to send e-mail messages.

- **PROFINET Station Name**

  The module needs to be assigned a Station Name in order to participate on PROFINET.

  Normally set from the network.

See also...

- "Identification & Maintenance (I&M)" on page 26
- "Web Server" on page 33
- "Network Configuration Object (04h)" on page 77
- "HICP (Host IP Configuration Protocol)" on page 163

# 2.4 Network Data Exchange

## 2.4.1 Application Data Instances (ADIs)

ADIs can be accessed acyclically from the network by means of Record Data read/write services. If addressed through a given API and Index range, the module translates the service into standard object requests towards the Application Data Object. If the host application responds with an error to such a request, that error will be translated to PROFINET standard.

The following parameters affect the addressing of ADIs on PROFINET:

- **Application Process Instance (API)**

  API 0 (zero) provides access to data in the Application Data Object, i.e. the ADIs. Acyclic requests towards other APIs will be forwarded to the PROFINET IO Object (F6h) by means of the 'Get_Record' and 'Set_Record'-commands.

  The remainder of this section assumes API 0 (zero).

- **Slot & subslot**

  The Slot and subslot values have no impact on the actual addressing of ADIs, except that the actual Slot and subslot needs to be populated with a module/submodule. This is always true for the DAP (Device Access Point), which occupies Slot #0/subslot #1. Other Slot/subslot values can also be used provided that the implementation populates it with a module/submodule.

- **Index**

  There is a 1:1 correlation between ADI and index as long as the index number is less than - or equal to - 7FFFh. Index 0 (zero) is not associated with an ADI and cannot be used.

| API | Slot | Subslot | Index | ADI | Comments |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0000h | - | (not associated with ADIs) |
| | | | 0001h | 1 | Device Access Point (DAP) |
| | | | 0002h | 2 | |
| | | | ... | ... | |
| | | | 7FFFh | 32767 | |
| | | | 8000h...FFFFh | - | (not associated with ADIs) |
| | X (>0) | Y | 0000h | | |
| | | | 0001h | 1 | Conditional; X and Y must be populated |
| | | | 0002h | 2 | |
| | | | ... | ... | |
| | | | 7FFFh | 32767 | |
| | | | 8000h...FFFFh | - | (not associated with ADIs) |
| >0 | - | - | - | - | See "Application Process Instances (API)" on page 22 |

See also...

- "Caveats" on page 17
- "Application Process Instances (API)" on page 22
- "PROFINET IO Object (F6h)" on page 130

### 2.4.2 Process Data

Mapping an ADI to Write Process Data results in PROFINET input data, and mapping an ADI to Read Process Data results in PROFINET output data. Consistency over all I/O data mapped on PROFINET is guaranteed.

See also...

- "Real Identification (RI)" on page 23

---

**IMPORTANT:** *The order in which ADIs are mapped to Process Data is significant and must be replicated in the IO Controller when setting up the network communication (i.e. modules must be set up in the same order, size and direction, as the mapped ADIs). See "Real Identification (RI)" on page 23 for more information.*

### 2.4.3 Caveats

The length parameter in the Record Data request specifies the number of bytes to read/write.

- When reading more data than the actual size of the ADI, the response will only contain the actual ADI data, i.e. no padding on the data is performed by the module.
- When writing to an ADI, the length parameter is not checked by the module, i.e. the host application must respond with an error if the length differs from the actual size of the requested ADI.

See also...

- "Application Process Instances (API)" on page 22

## 2.5 Web Interface

The built-in web server can be used to provide rich, dynamic content, by means of JSON and SSI scripting. This enables access to information and configuration settings within the file system, as well as through the Anybus CompactCom object module.

Web server content resides within the module's file system. This means that it can be accessed and customized as needed using a standard FTP client.

See also...

- "File System" on page 19
- "FTP Server" on page 31
- "Web Server" on page 33
- "Server Side Include (SSI)" on page 40

# 2.6 E-mail Client

The built-in e-mail client enables the host application to send e-mail messages stored in the file system, or defined directly within the SMTP Client Object (09h). Messages are scanned for SSI content, which means it's possible to embed dynamic information from the file system or from the Anybus CompactCom 40 object model.

See also...

- 2-19 "File System"
- 6-39 "E-mail Client"
- 7-40 "Server Side Include (SSI)"
- 11-118 "SMTP Client Object (09h)"

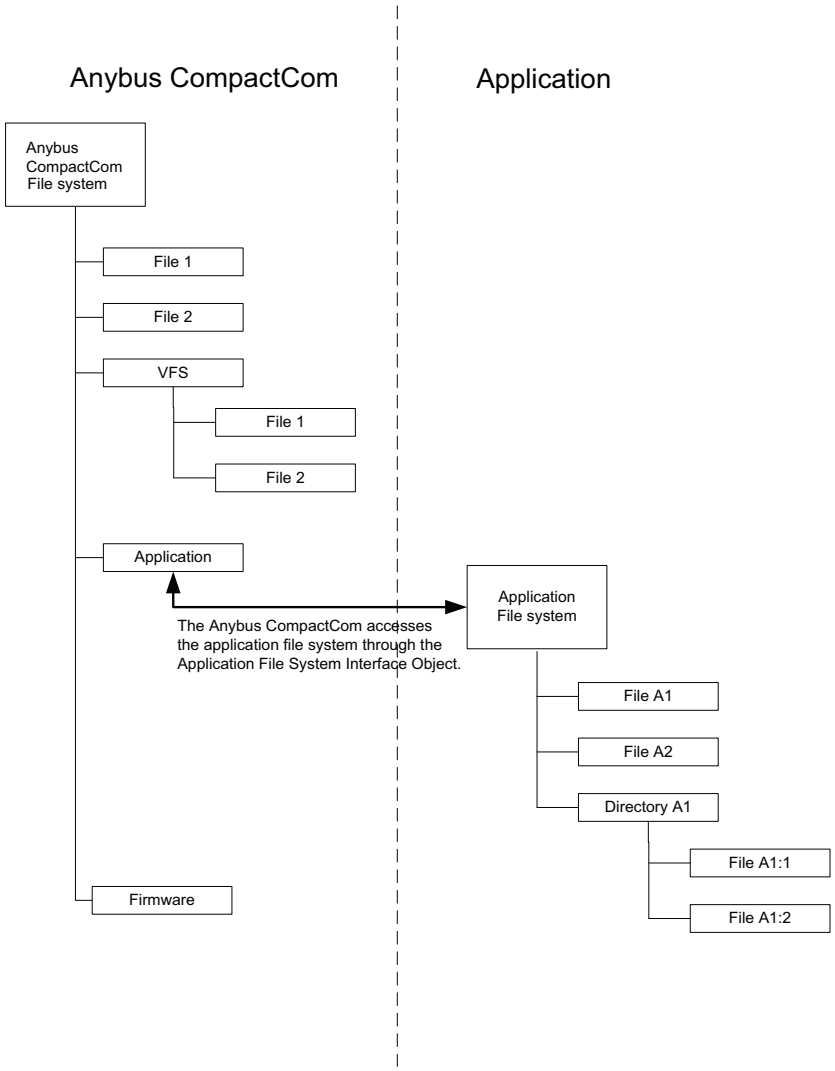
# 2.7 File System

## 2.7.1 Overview

The Anybus CompactCom 40 PROFINET IO has a built-in file system, that can be accessed from the application and from the network. Three directories are predefined:

- VFS - The virtual file system that e.g. holds the web pages of the module.
- Application - This directory provides access to the application file system through the Application File System Interface Object (EAh) (optional).
- Firmware - Firmware updates are stored in this directory.

**Important**: In the firmware folder, it is not possible to use append mode when writing a file. Be sure to use write mode only.



## 2.7.2 General Information

The built-in file system hosts 28 Mb of nonvolatile storage, which can be accessed by the HTTP and FTP servers, the e-mail client, and the host application (through the Anybus File System Interface Object (0Ah).

The file system uses the following conventions:

- '\' (backslash) is used as a path separator
- Names may contain spaces (' ') but must not begin or end with one.
- Valid characters in names are ASCII character numbers less than 127, excluding the following characters: '\ / : * ? " < > |'
- Names cannot be longer than 48 characters
- A path cannot be longer than 126 characters (filename included)

See also...

- "FTP Server" on page 31
- "Web Server" on page 33
- "E-mail Client" on page 39
- "Server Side Include (SSI)" on page 40
- "File System Interface Object (0Ah)" on page 123

---

**IMPORTANT:** *The file system is located in flash memory. Due to technical reasons, each flash segment can be erased approximately 100000 times before failure, making it unsuitable for random access storage.*

*The following operations will erase one or more flash segments:*

- *Deleting, moving or renaming a file or directory*
- *Writing or appending data to an existing file*
- *Formatting the file system*

## 2.7.3 System Files

The file system contains a set of files used for system configuration. These files, known as "system files", are regular ASCII files which can be altered using a standard text editor (such as the Notepad in Microsoft Windows™). The format of these files are, with some exceptions, based on the concept of 'keys', where each 'key' can be assigned a value, see below.

Example:

```
[Key1]
value of Key1

[Key2]
value of Key2
```

# 3. PROFINET IO Implementation Details

## 3.1 General Information

This chapter covers PROFINET specific details in the Anybus implementation. Note that the use of such functionality may require in-depth knowledge in PROFINET networking internals and/or information from the official PROFINET specification. In such cases, the people responsible for the implementation of this product are expected either to obtain these specifications to gain sufficient knowledge or limit their implementation in such a way that this is not necessary. The GSD file must be changed to reflect all changes.

Implementation overview:

- **Conformance Class**

  The Anybus module complies to conformance class C.

- **Performance Characteristics**
  - 100 Mbps, full duplex with autonegotiation enabled as default
  - Real Time (RT) communication, 250 µs cycle time
  - Isochronous Real Time (IRT) communication, 250 µs cycle time

- **Device Model**
  - One IO Device instance
  - The IO Device instance includes an Application Process referenced by its identifier (API). API 0 (zero) is implemented by default.
  - The API implements one or more slots
  - Each Slot implements one or more subslots
  - Each subslot may implement one or more Channels

- **Slots & Subslots**

  Up to 128 subslots in total.

- **IO Data**

  1440 bytes of IO data in each direction, including status bytes (4 bytes for DAP submodules + 1 byte per additional submodule)".

See also...

- "Electronic Data Sheet (GSD)" on page 13

## 3.2 Application Process Instances (API)

As mentioned previously, acyclic requests towards API 0 are forwarded to the Application Data Object.

Cyclic data exchange is by default carried out through API 0 (i.e. the Anybus associates modules and submodules with API 0).

See also...

- "Real Identification (RI)" on page 23

## 3.3 Application Relationships (AR)

On PROFINET, a connection between an IO Controller/Supervisor and an I/O device (in this case the Anybus) is called 'Application Relationship' (AR). The Anybus module supports multiple simultaneous Application Relationships, allowing multiple IO Supervisors to access its data and functions.

The host implementation can either ignore this functionality altogether, in which case the Anybus module will handle it automatically, or integrate the establishment and handling of Application Relationships into the host firmware.

Application Relationships are managed through the following functions:

- AR_Check_Ind (see "Command Details: AR_Check_Ind" on page 138)
- Expected_Ident_Ind (see "Command Details: Expected_Ident_Ind" on page 140)
- AR_Abort_Ind (see "Command Details: AR_Abort_Ind" on page 143)
- AR_Abort (see "Command Details: AR_Abort" on page 97)

# 3.4 Real Identification (RI)

## 3.4.1 General Information

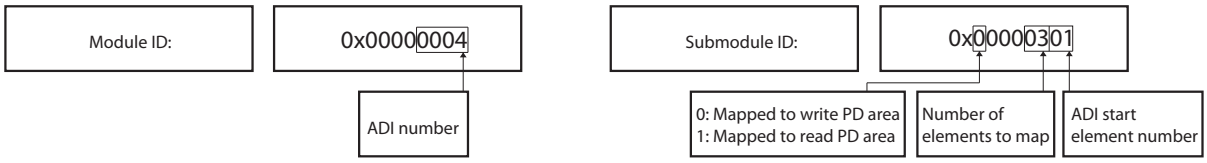
During establishment of an IO Connection towards the Anybus module, the configuration derived from the IO Controller (i.e. the 'Expected Identification') and the actual configuration in the Anybus module (i.e. the 'Real Identification' or RI) are compared.

- **Default Configuration (ADI Based)**

  By default (i.e. if the application doesn't issue API_Add, Plug_Module, Plug_Submodule), the Anybus handles the plugging of modules and submodules automatically in accordance with the mapped Process Data as follows:

  - A DAP is plugged into Slot 0 (zero)
  - Modules are added in consecutive order (based on the order of the mapping commands)
  - All modules belong to API 0 (zero)

  The module identification number and the submodule identification number should be interpreted as follows:




Example (100BASE-TX DAP):

| ADI # | Type | Resulting Real Identification | | |
|---|---|---|---|---|
| | | Module/Submodule ID | Slot/Subslot | IO Data Size (bytes) |
| - | - | 0x80010000/0x00000001<br>0x80010000/0x00000002<br>0x80010000/0x00000003<br>0x80010000/0x00000003 | 0/1 (Device Access Point)<br>0/0x8000 (Interface)<br>0/0x8001 (Port1)<br>0/0x8002 (Port2) | - |
| 6 | 1 SINT16 Wr | 0x00000006/0x00000100 | 1/1 | 2/0 |
| 3 | 2 UINT8 Rd | 0x00000003/0x10000200 | 2/1 | 0/2 |
| 3 | 1 UINT8 Rd | 0x00000003/0x10000104 | 2/2 | 0/1 |
| 9 | 1 SINT32 Rd | 0x00000009/0x10000100 | 3/1 | 0/4 |
| 10 | 1 SINT16 Wr | 0x0000000A/0x00000100 | 4/1 | 2/0 |

See also...

- "Application Process Instances (API)" on page 22

### 3.4.2 Configuration Mismatch

If the application has implemented support for the remap commands in the Application Data object (FEh), the Anybus module will, in case of a configuration mismatch using the ADI mode, try to remap the Real Identification to match the Expected Identification.

As the Module ID contains the ADI number and the Submodule ID describes I/O direction and element section, all information required to perform a remap is available.

**Note**: The application must be able to respond with the data type for every ADI, or a remap is not possible.

The remap is performed in two steps, first the read area is remapped and then the write area. If the first remap request is rejected, no change to the process data map and Real Identification is done. If the second remap request is rejected, the process data map is in an inconsistent state and new remap commands are sent that will remove all mappings. If both remap requests succeed a new Real Identification will be built to match the Expected Identification.

A controller may choose to connect to fewer submodules than are plugged by the application (plugged explicitly using Plug_Module/Submodule or implicitly by mapping ADIs to process data).

- **Incomplete output mapping:**

  If the controller tries to connect to fewer output submodules than are plugged by the application, the controller will not be granted ownership of any output submodule. Anybus state is set to ERROR and LEDs will indicate "Config mismatch". Matching input submodules will be owned by controller and work normally.

- **Incomplete input mapping:**

  The controller may choose to connect to a subset of the available input submodules without any restrictions. Anybus state is set to PROCESS_ACTIVE and no error is indicated on LED.

- **Mismatch of submodule(s):**

  As long as all of the output submodules of the Real Identification are present and matching in the Expected Identification, the Anybus state is set to PROCESS_ACTIVE. However, if there is any mismatch among the other submodules the LEDs will indicate "Config mismatch".

**Note**: In the case of ADI based Real Identification mode, a remap is first attempted for cases 1 and 3 above.

See also...

- The Remap_ADI_Write_Area and Remap_ADI_Read_Area commands in the Application Data Object (FEh), found in Anybus CompactCom 40 Software Design Guide.

# 3.5 Diagnostics

## 3.5.1 Standard Diagnostics

PROFINET IO uses alarms when informing the IO Controller of diagnostic entries. In the Anybus implementation, it is possible for the application to create alarms via diagnostic entries by means of the Diagnostic Object (02h).

Up to 5 diagnostic instances can be created by the host application. An additional 6th instance can always be created in event of a major unrecoverable fault.

Creating a diagnostic instance is done by issuing the command Create. If the module is in state IDLE or PROCESS_ACTIVE, the created instance will be communicated on the network as an "appear"-alarm. If the module is in another state, the PLC will be notified in the connect response by a module diff block.

Deleting a diagnostic instance is done by issuing the command Delete. This will trigger a "disappear"-alarm on the network. Supply the instance ID that was returned by the create-command.

Every diagnostic instance has a severity level and an event code associated to it. Major unrecoverable events will cause the module to disconnect itself from the network, thus preventing network participation. Other severity levels either produce a Channel Diagnostic alarm or a Generic Diagnostic alarm, depending on the Event Code, according to the table below.

| Severity | Event code != network specific | Event code = network specific |
|---|---|---|
| Minor, recoverable | Channel Diagnostic Alarm | Generic Diagnostic Alarm<br>(See "Extended Diagnostics" on page 25) |
| Minor, unrecoverable | | |
| Major, recoverable | | |
| Major, unrecoverable | Anybus enters Exception state | |

**Note**: Process alarms can not be created.

See also...

- "Diagnostic Object (02h)" on page 73.

## 3.5.2 Extended Diagnostics

Using the network specific event code (FFh) creates a Generic Diagnostic Alarm on the network. This type of alarm can carry extended diagnostic information and more details about the source of the problem.

Generic Diagnostic Alarm instances can be tagged with a source API and slot- and subslot number, and can also contain additional network specific diagnostic data.

For more information, see "Details: Network Specific Data" on page 75.

# 3.6 Identification & Maintenance (I&M)

## 3.6.1 General Information

Identification & Maintenance (I&M) provides a standard way of gathering information about an I/O device. The I&M information can be accessed by the IO Controller by means of acyclic Record Data Read/Write services.

The application should provide application specific I&M0 information during start-up. See "PROFINET IO Object (F6h)" on page 130 for more information.

It is possible for the application to handle I&M records. Activate this using the IM_Options command. See "Network PROFINET IO Object (0Eh)" on page 85 for more information.

Default I&M0 information:

| | |
|---|---|
| **IM Manufacturer ID** | 010Ch (HMS Industrial Networks) |
| **IM Order ID** | 'ABCC40-PIR' |
| **IM Serial Number** | (unique serial number, set during manufacturing) |
| **IM Hardware Revision** | (Anybus hardware revision ID, set during manufacturing) |
| **IM Software Revision** | (Anybus software revision, set during manufacturing) |
| **IM Revision Counter** | (Revision counter) |
| **IM Profile ID** | F600h (Generic Device) |
| **IM Profile Specific Type** | 0004h (No profile) |
| **IM Version** | 0101h |
| **IM Supported** | 001Eh (IM0-4 supported) |

### 3.6.2 I&M Data Structures

The I&M records uses the following data structures.

| Record | Content | Size | Description |
|---|---|---|---|
| I&M0 | Manufacturer Id | 2 bytes | PROFINET IO Object (F6h), attribute #2 ('Vendor ID/I&M Vendor ID') |
| | Order Id | 20 bytes | PROFINET IO Object (F6h), attribute #8 ('I&M Order ID') |
| | Serial number | 16 bytes | PROFINET IO Object (F6h), attribute #9 ('I&M Serial number') |
| | Hardware revision | 2 bytes | PROFINET IO Object (F6h), attribute #10 ('I&M Hardware revision') |
| | Software revision | 4 bytes | PROFINET IO Object (F6h), attribute #11 ('I&M Software revision') |
| | Revision counter | 2 bytes | PROFINET IO Object (F6h), attribute #12 ('I&M Revision counter') |
| | Profile Id | 2 bytes | PROFINET IO Object (F6h), attribute #13 ('I&M Profile ID') |
| | Profile specific type | 2 bytes | PROFINET IO Object (F6h), attribute #14 ('I&M Profile specific type') |
| | IM version | 2 bytes | 0101h (Internal, constant value) |
| | IM supported | 2 bytes | 001Eh (Internal, constant value) |
| I&M1 | Tag Function | 32 bytes | Default: All bytes set to blanks (' ') |
| | Tag Location | 22 bytes | Default: All bytes set to blanks (' ') |
| I&M2 | Installation date | 16 bytes | Default: All bytes set to blanks (' ') |
| I&M3 | Descriptor | 54 bytes | Default: All bytes set to blanks (' ') |
| I&M4 | Signature | 54 bytes | Default: All bytes set to zero (00h) |

See also...

- "PROFINET IO Object (F6h)" on page 130

# 3.7 Fast Start Up

## 3.7.1 General Information

The Fast Start Up (FSU) function enables PROFINET IO devices, connected to the network, to power up quickly. This is useful in, for example, robot applications, where rapid retooling is necessary. With FSU activated, the module will send a DCP Hello message as soon as possible after power-on.

This function is enabled by two GSD keywords: PowerOnToCommReady and DCP_HelloSupported. The activation is made from the PLC configuration tool.

The FSU time is defined as the number of milliseconds (ms) from hardware reset (or power-on) until the module enters the PROCESS_ACTIVE state. On PROFINET, it is recommended to try to reach a FSU time <= 500 ms.

To enable FSU, set values according to the following (listed for the Device Access Point(s)):

- **PowerOnToCommReady**
  FSU time, in milliseconds (ms). This value must be measured by the customer.
- **DCP_HelloSupported**
  Value: true.

To disable FSU, set the keywords to the following values:

- **PowerOnToCommReady**
  Remove this keyword from the GSD file.
- **DCP_HelloSupported**
  Value: false.

## 3.7.2 How to Improve the FSU Time

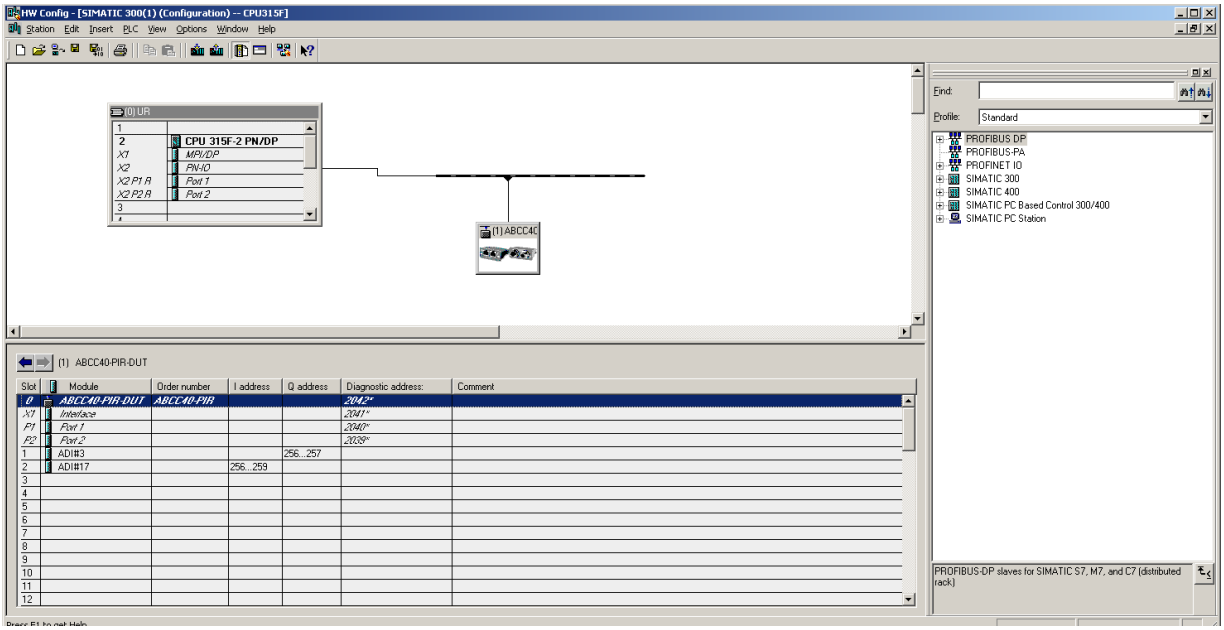
- Issue as few commands as possible to the module during the SETUP state.
- Respond as fast as possible to all commands issued by the module after setup is complete.
- If there is record data in the GSD file, use attribute #4 in the Application Object (FFh), instance #1 (Parameter Control Sum). During the next start-up, this parameter data is already saved in the module's nonvolatile storage and will not be sent to the application.

### 3.7.3 Fast Start Up Configuration with STEP7

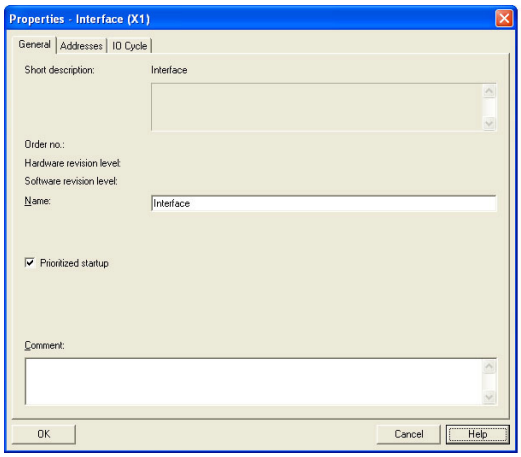
The example below shows the procedure when the Siemens tool STEP7 is used for configuration.

#### Activation of Fast Start Up

1. Start the configuration tool. The figure below shows the HW Config window of the STEP7 tool.

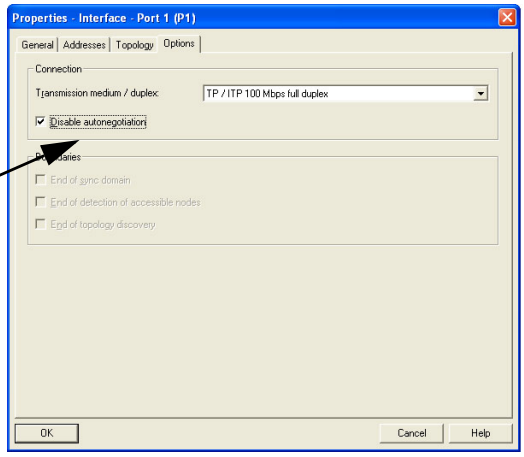


**2.** Double click on "Interface" in the Module column. The window shown to the right will appear. Choose the "General" tab and check the box "Prioritized startup".




**3.** Return to the HW Config window. Double click on "Port 1" in the Module column. The window shown to the right will appear. Choose the Options tag. To configure fastest possible startup, choose transmission medium/duplex "TP/ITP 100 Mbps, full duplex" and check the "Disable autonegotiation" box.

**4.** Repeat for Port 2.




# 3.8 Address Conflict Detection (ACD)

The Anybus CompactCom 40 PROFINET IO module supports Address Conflict Detection (ACD). This mechanism involves the following two aspects:

- Initial probing: before using an IP address, the module issues ARP probes to see if the address is already in use (three probes with a 100 ms delay).
- Address announcement: after the initial probing, the module issues ARP announcements.

If an IP address conflict is detected, "IP address error" will be indicated on the Network Status LED. The module will use address 0.0.0.0. A new address can be configured via the Anybus IPconfig tool.

**IMPORTANT**: If Fast Start Up is used, ACD initial probing is automatically disabled to ensure a fast startup. Address announcement is still used, as it will not affect the actual startup time.

To enable/disable ACD, see "Ethernet Host Object (F9h)" on page 145.

# 4. FTP Server

## 4.1 General Information

**Category**: extended

The built-in FTP-server makes it easy to manage the file system using a standard FTP client.

The following port numbers are used for FTP communication:

- TCP, port 20 (FTP data port)
- TCP, port 21 (FTP command port)

The FTP server supports up to 8 concurrent connections.

## 4.2 User Accounts

User accounts are stored in the configuration file '\ftp.cfg'. This file holds the usernames, passwords, and home directory for all users. Users are not able to access files outside of their home directory.

File Format:

```
User1:Password1:Homedir1
User2:Password2:Homedir2
User3:Password3:Homedir3
```

Optionally, the UserN:PasswordN-section can be replaced by a path to a file containing a list of users as follows:

File Format ('\ftp.cfg'):

```
User1:Password1:Homedir1
User2:Password2:Homedir2
\path\userlistA:HomedirA
\path\userlistB:HomedirB
```

The files containing the user lists shall have the following format:

File Format:

```
User1:Password1
User2:Password2
User3:Password3
```

**Notes:**

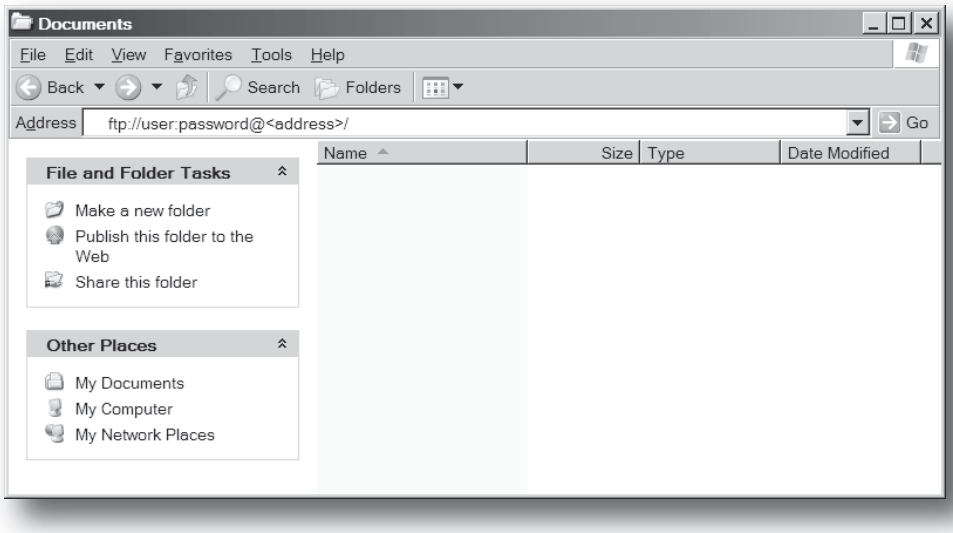
- Usernames must not exceed 15 characters in length.
- Passwords must not exceed 15 characters in length.
- Usernames and passwords must only contain alphabetic characters and/or numbers.
- If '\ftp.cfg' is missing or cannot be interpreted, all username/password combinations will be accepted and the home directory will be the FTP root (i.e. '\ftp\').
- The home directory for a user must also exist in the file system if they should be able to log in, just adding the user information to the 'ftp.cfg' file it is not enough.

- If 'Admin Mode' has been enabled in the Ethernet Object, all username/password combinations will be accepted and the user will have unrestricted access to the file system (i.e. the home directory will be the system root).
- It is strongly recommended to have at least one user with root access ('\') permission. If not, 'Admin Mode' must be enabled each time a system file needs to be altered (including '\ftp.cfg').

## 4.3 Session Example

The Windows Explorer features a built-in FTP client which can easily be used to access the file system as follows:

1. Open the Windows Explorer by right-clicking on the 'Start'-button and selecting 'Explorer'
2. In the address field, type FTP://<user>:<password>@<address>
   - Substitute <address> with the IP address of the Anybus module
   - Substitute <user> with the username
   - Substitute <password> with the password
3. Press enter. The Explorer will now attempt to connect to the Anybus module using the specified settings. If successful, the file system will be displayed in the Explorer window.



# 5. Web Server

## 5.1 General Information

**Category**: extended

The built-in web server provides a flexible environment for end-user interaction and configuration purposes. The powerful combination of SSI and client-side scripting allows access to objects and file system data, enabling the creation of advanced graphical user interfaces.

The web interfaces is stored in the file system, which can be accessed through the FTP server. If necessary, the web server can be completely disabled in the Ethernet Host Object.

The web server supports up to 20 concurrent connections and communicates through port 80.

See also...

- "FTP Server" on page 31
- "Server Side Include (SSI)" on page 40
- "JSON" on page 58
- "Ethernet Host Object (F9h)" on page 145

## 5.2 Default Web Pages

The default web pages provide access to:

- Network configuration parameters
- Network status information
- Access to the host application ADIs

The default web pages are built of files stored in a virtual file system accessible through the vfs folder. These files are read only and cannot be deleted or overwritten. The web server will first look for a file in the web root folder. If not found it will look for the file in the vfs folder, making it appear as the files are located in the web root folder. By loading files in the web root folder with exactly the same names as the default files in the vfs folder, it is possible to customize the web pages, replacing such as pictures, logos and style sheets.

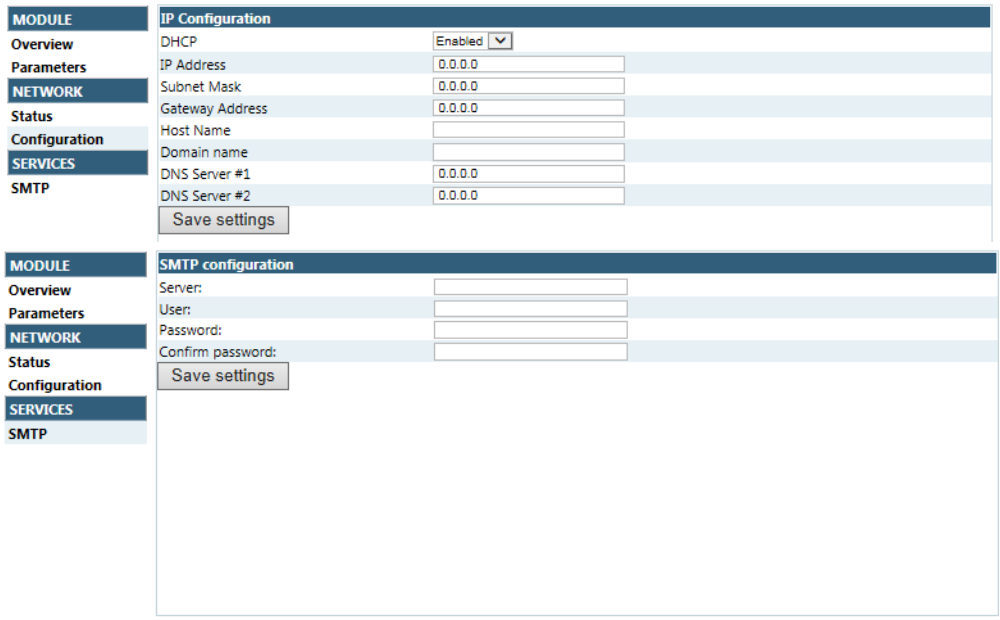
If a complete customized web system is designed and no files in the vfs folder are to be used, it is recommended to turn off the virtual file system completely, see the File System Interface Object.

See also...

- "File System" on page 19
- "File System Interface Object (0Ah)" on page 123

### 5.2.1 Network Configuration

The network configuration page provides interfaces for changing TCP/IP and SMTP settings in the Network Configuration Object.




The module needs a reset for the changes to take effect.

#### Available IP Configuration Settings

| Name | Description |
|---|---|
| DHCP | Checkbox for enabling or disabling DHCP<br>Default value: disabled |
| IP address | The TCP/IP settings of the module<br>Default values: 0.0.0.0<br>Value ranges: 0.0.0.0 - 255.255.255.255 |
| Subnet mask | |
| Gateway address | |
| Host name | IP address or name<br>Max 64 characters |
| Domain name | IP address or name<br>Max 48 characters |

#### Available SMTP Settings

| Name | Description |
|---|---|
| Server | IP address or name<br>Max 64 characters |
| User | Max 64 characters |
| Password | Max 64 characters |

### 5.2.2 Ethernet statistics page

The Ethernet statistics web page contains the following information:

| Current IP Configuration | Description |
|---|---|
| DHCP: | - |
| Host Name: | - |
| IP Address: | - |
| Subnet Mask: | - |
| Gateway Address: | - |
| DNS Server #1: | - |
| DNS Server #2: | - |
| Domain Name: | - |

| Current Ethernet Configuration | | Description |
|---|---|---|
| MAC Address | | - |
| Port 1 | Speed: | The current link speed. |
| | Duplex: | The current duplex configuration. |
| Port 2 | Speed: | The current link speed. |
| | Duplex: | The current duplex configuration. |

| Interface Counters | Description |
|---|---|
| In Octets: | Received bytes. |
| In Ucast Packets: | Received unicast packets. |
| In NUcast packets: | Received non-unicast packets (broadcast and multicast). |
| In Discards: | Received packets discarded due to no available memory buffers. |
| In Errors: | Received packets discarded due to reception error. |
| In Unknown Protos: | Received packets with unsupported protocol type. |
| Out Octets: | Sent bytes. |
| Out Ucast packets: | Sent unicast packets. |
| Out NUcast packets: | Sent non-unicast packets (broadcast and multicast). |
| Out Discards: | Outgoing packets discarded due to no available memory buffers. |
| Out Errors: | Transmission errors. |

| Media Counters | Description |
|---|---|
| Alignment Errors | Frames received that are not an integral number of octets in length. |
| FCS Errors | Frames received that do not pass the FCS check. |
| Single Collisions | Successfully transmitted frames which experienced only one collision. |
| Multiple Collisions | Successfully transmitted frames that experienced more than one collision. |
| SQE Test Errors | Number of times SQE test error messages are generated. |
| Deferred Transmissions | Frames for which first transmission attempt is delayed because the medium is busy. |
| Late Collisions | Number of times a collision is detected later than 512 bit-times into the transmission of a packet. |
| Excessive Collisions | Frames for which a transmission fails due to excessive collisions. |
| MAC Receive Errors | Frames for which reception of an interface fails due to an internal MAC sublayer receive error. |
| MAC Transmit Errors | Frames for which transmission fails due to an internal MAC sublayer receive error. |
| Carrier Sense Errors | Times that the carrier sense condition was lost or never asserted when attempted to transmit a frame. |
| Frame Size Too Long | Frames received that exceed the maximum permitted frame size. |
| Frame Size Too Short | Frames received that are shorter than lowest permitted frame size. |

| Fiber Optical Statistics[a] | Description |
|---|---|
| Port 1 Temperature (C): | Current temperature of port 1 transceiver, in degrees Celsius. |
| Port 1 Power Budget (dB): | Current received power budget for port 1 transceiver, in dB. |
| Port 1 Power Budget Status: | Textual display of the power budget status for port 1:<br>“OK”: Power budget status > 2 dB<br>“Maintenance required”: 0 dB < power budget status < 2 dB<br>“Maintenance demanded”: Power budget = 0 dB |
| Port 2 Temperature (C): | Current temperature of port 2 transceiver, in degrees Celsius. |
| Port 2 Power Budget (dB): | Current received power budget for port 2 transceiver, in dB. |
| Port 2 Power Budget Status: | Textual display of the power budget status for port 2:<br>“OK”: Power budget status > 2 dB<br>“Maintenance required”: 0 dB < power budget status < 2 dB<br>“Maintenance demanded”: Power budget = 0 dB |

a. These statistics are only available for the PROFINET IRT FO module.

# 5.3 Server Configuration

## 5.3.1 General Information

**Category**: advanced

Basic web server configuration settings are stored in the system file '\http.cfg'. This file holds the root directory for the web interface, content types, and a list of file types which shall be scanned for SSI.

*File Format:*

```
[WebRoot]
\web

[FileTypes]
FileType1:ContentType1
FileType2:ContentType2
...
FileTypeN:ContentTypeN

[SSIFileTypes]
FileType1
FileType2
...
FileTypeN
```

**Web Root Directory**

The web server cannot access files outside this directory.

**Content Types**

A list of file extensions and their reported content types.

See also...

- "Default Content Types" on page 38

**SSI File Types**

By default, only files with the extension 'shtm' are scanned for SSI. Additional SSI file types can be added here as necessary.

The web root directory determines the location of all files related to the web interface. Files outside of this directory and its subdirectories *cannot* be accessed by the web server.

## 5.3.2 Index Page

The module searches for possible index pages in the following order:

1. <WebRoot>\index.htm
2. <WebRoot>\index.html
3. <WebRoot>\index.shtm
4. <WebRoot>\index.wml

**Note 1:** Substitute <WebRoot> with the web root directory specified in '\http.cfg'.

**Note 2:** If no index page is found, the module will default to the virtual index file (if enabled).

See also...

- "Default Web Pages" on page 33

### 5.3.3 Default Content Types

By default, the following content types are recognized by their file extension:

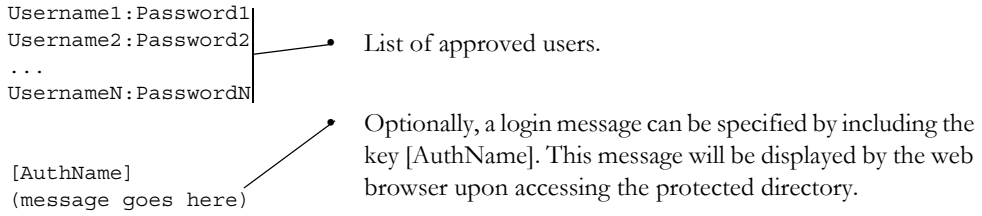
| File Extension | Reported Content Type |
|---|---|
| htm, html, shtm | text/html |
| gif | image/gif |
| jpeg, jpg, jpe | image/jpeg |
| png | image/x-png |
| js | application/x-javascript |
| bat, txt, c, h, cpp, hpp | text/plain |
| zip | application/x-zip-compressed |
| exe, com | application/octet-stream |
| wml | text/vnd.wap.wml |
| wmlc | application/vnd.wap.wmlc |
| wbmp | image/vnd.wap.wbmp |
| wmls | text/vnd.wap.wmlscript |
| wmlsc | application/vnd.wap.wmlscriptc |
| xml | text/xml |
| pdf | application/pdf |
| css | text/css |

Content types can be added or redefined by adding them to the server configuration file, see "General Information" on page 37.

### 5.3.4 Authorization

Directories can be protected from web access by placing a file called 'web_accs.cfg' in the directory to protect. This file shall contain a list of users that are allowed to access the directory and its subdirectories.

File Format:




The list of approved users can optionally be redirected to one or several other files.

**Note:** If the list of approved users is put in another file, be aware that this file can be accessed and read from the network.

Example:

In this example, the list of approved users will be loaded from 'here.cfg' and 'too.cfg'.

```
[File path]
\i\put\some\over\here.cfg
\i\actually\put\some\of\it\here\too.cfg

[AuthName]
Howdy. Password, please.
```

# 6. E-mail Client

## 6.1 General Information

**Category**: extended

The built-in e-mail client allows the application to send e-mail messages through an SMTP-server. Messages can either be specified directly in the SMTP Client Object, or retrieved from the file system. The latter may contain SSI, however note that for technical reasons, certain commands cannot be used (specified separately for each SSI command).

The client supports authentication using the 'LOGIN' method. Account settings etc. are stored in the Network Configuration Object.

See also...

- "Network Configuration Object (04h)" on page 77
- "SMTP Client Object (09h)" on page 118

## 6.2 How to Send E-mail Messages

To be able to send e-mail messages, the SMTP-account settings must be specified.

This includes...

- A valid SMTP-server address
- A valid username
- A valid password

To send an e-mail message, perform the following steps:

1. Create a new e-mail instance using the 'Create'-command (03h)
2. Specify the sender, recipient, topic and message body in the e-mail instance
3. Issue the 'Send Instance Email'-command (10h) towards the e-mail instance
4. Optionally, delete the e-mail instance using the 'Delete'-command (04h)

Sending a message based on a file in the file system is achieved using the 'Send Email from File'-command. For a description of the file format, see "Command Details: Send E-mail From File" on page 121.

# 7. Server Side Include (SSI)

## 7.1 General Information

**Category**: advanced

Server Side Include functionality, or SSI, allows data from files and objects to be represented on web pages and in e-mail messages.

SSI are special commands embedded within the source document. When the Anybus CompactCom module encounters such a command, it will execute it, and replace it with the result (if applicable).

By default, only files with the extension 'shtm' are scanned for SSI.

## 7.2 Include File

This function includes the contents of a file. The content is scanned for SSI.

**Note:** This function cannot be used in e-mail messages.

Syntax:

```
<?--#include file="filename"-->
```

filename-Source file

Default Output:

| Scenario | Default Output |
|---|---|
| Success | (contents of file) |

# 7.3 Command Functions

## 7.3.1 General Information

Command functions executes commands and includes the result.

General Syntax:

```
<?--#exec cmd_argument='command'-->
```

command-Command function, see below.

Command Functions:

| Command | Valid for E-mail Messages | Page |
|---|---|---|
| GetConfigItem() | Yes | 42 |
| SetConfigItem() | No | 43 |
| SsiOutput() | Yes | 45 |
| DisplayRemoteUser | No | 45 |
| ChangeLanguage() | No | 46 |
| IncludeFile() | Yes | 47 |
| SaveDataToFile() | No | 48 |
| printf() | Yes | 49 |
| scanf() | No | 51 |

### 7.3.2 GetConfigItem()

This command returns specific information from a file in the file system.

File Format:

The source file must have the following format:

```
[key1]
value1

[key2]
value2
...
[keyN]
valueN
```

Syntax:

```
<?--exec cmd_argument='GetConfigItem("filename", "key"[,"separator"])'-->
```

filename- Source file to read from.
key - Source [key] in file.
separator- Optional; specifies line separation characters (e.g. "<br>"). (default is CRLF).

Default Output:

| Scenario | Default Output |
|---|---|
| Success | *(value of specified key)* |
| Authentication Error | "Authentication error " |
| File open error | "Failed to open file "*filename*" " |
| Key not found | "Tag (*key*) not found " |

Example:

The following SSI...

```
<?--exec cmd_argument='GetConfigItem("\fruit.cnf", "Lemon")'-->
```

... in combination with the following file ('\fruit.cnf')...

```
[Apple]
Green

[Lemon]
Yellow

[Banana]
Blue
```

... returns the string 'Yellow'.

### 7.3.3 SetConfigItem()

This function stores an HTML-form as a file in the file system.

**Note:** This function cannot be used in e-mail messages.

Syntax:

```
<?--#exec cmd_argument='SetConfigItem("filename" [, Overwrite])'-->
```

| | |
|---|---|
| filename- | Destination file. If the specified file does not exist, it will be created (provided that the path is valid). |
| Overwrite | -Optional; forces the module to create a new file each time the command is issued. The default behavior is to modify the existing file. |

File Format:

Each form object is stored as a [tag], followed by the actual value.

```
[form object name 1]
form object value 1

[form object name 2]
form object value 2

[form object name 3]
form object value 3

...

[form object name N]
form object value N
```

**Note:** Form objects with names starting with underscore ('_') will not be stored.

Default Output:

| Scenario | Default Output |
|---|---|
| Success | "Configuration stored to "*filename*" " |
| Authentication Error | "Authentication error " |
| File open error | "Failed to open file "*filename*" " |
| File write error | "Could not store configuration to *"filename"* " |

Example:

The following example demonstrates how to use this function. The resulting page sends a form to itself, which is then evaluated by the SetConfigItem command.

```
<HTML>
<HEAD><TITLE>SetConfigItem Test</TITLE></HEAD>
<BODY>

<?--#exec cmd_argument='SetConfigItem("\food.txt")'-->

<FORM action="test.shtm">
  <P>
    <LABEL for="Name">Name: </LABEL><BR>
    <INPUT type="text" name="Name"><BR><BR>

    <LABEL for="_Age">Age: </LABEL><BR>
    <INPUT type="text" name="_Age"><BR><BR>

    <LABEL for="Food">Food: </LABEL><BR>
    <INPUT type="radio" name="Food" value="Cheese"> Cheese<BR>
    <INPUT type="radio" name="Food" value="Sausage"> Sausage<BR><BR>

    <LABEL for="Drink">Drink: </LABEL><BR>
    <INPUT type="radio" name="Drink" value="Wine"> Wine<BR>
    <INPUT type="radio" name="Drink" value="Beer"> Beer<BR><BR>

    <INPUT type="submit" name="_submit">
    <INPUT type="reset" name="_reset">
  </P>
</FORM>

</BODY>
</HTML>
```

The resulting file ('\food.txt') may look somewhat as follows:

```
[Name]
Cliff Barnes

[Food]
Cheese

[Drink]
Beer
```

**Note:** In order for this example to work, the HTML file must be named 'test.shtm'.

### 7.3.4 SsiOutput()

This command temporarily modifies the SSI output of the following command function.

Syntax:

```
<?--#exec cmd_argument='SsiOutput("success", "failure")'-->
```

success- String to use in case of success
failure - String to use in case of failure

Default Output:

(this command produces no output on its own)

Example:

The following example illustrates how to use this command.

```
<?--#exec cmd_argument='SsiOutput ("Parameter stored", "Error")'-->
<?--#exec cmd_argument='SetConfigItem("File.cfg", Overwrite)'-->
```

See also...

- "SSI Output Configuration" on page 57

### 7.3.5 DisplayRemoteUser

This command stores returns the username on an authentication session.

**Note:** This command cannot be used in e-mail messages.

Syntax:

```
<?--#exec cmd_argument='DisplayRemoteUser'-->
```

Default Output:

| Scenario | Default Output |
|---|---|
| Success | (current user) |

### 7.3.6 ChangeLanguage()

This command changes the language setting based on an HTML form object.

**Note:** This command cannot be used in e-mail messages.

Syntax:

```
<?--#exec cmd_argument='ChangeLanguage( "source" )'-->
```

source -Name of form object which contains the new language setting.
The passed value must be a single digit as follows:

| Form value | Language |
|---|---|
| "0" | English |
| "1" | German |
| "2" | Spanish |
| "3" | Italian |
| "4" | French |

Default Output:

| Scenario | Default Output |
|---|---|
| Success | "Language changed" |
| Error | "Failed to change language " |

Example:

The following example demonstrates how to use this function. The resulting page sends a form to itself, which is then evaluated by the ChangeLanguage() command.

```
<HTML>
<HEAD><TITLE>ChangeLanguage Test</TITLE></HEAD>
<BODY>

<?--#exec cmd_argument='ChangeLanguage("lang")'-->

<FORM action="test.shtm">
  <P>
    <LABEL for="lang">Language(0-4): </LABEL><BR>
    <INPUT type="text" name="lang"><BR><BR>

    <INPUT type="submit" name="_submit">
  </P>
</FORM>

</BODY>
</HTML>
```

**Note:** In order for this example to work, the HTML file must be named 'test.shtm'.

### 7.3.7 IncludeFile()

This command includes the content of a file. Note that the content is not scanned for SSI.

Syntax:

```
<?--#exec cmd_argument='IncludeFile("filename" [, separator])'-->
```

filename- Source file

separator- Optional; specifies line separation characters (e.g. "<br>").

Default Output:

| Scenario | Default Output |
|---|---|
| Success | *(file contents)* |
| Authentication Error | "Authentication error " |
| File open error | "Failed to open file "*filename*" " |

Example:

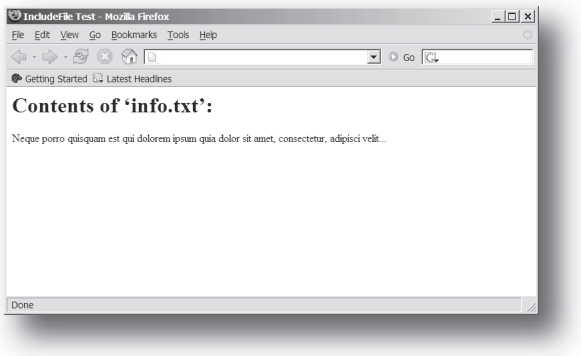
The following example demonstrates how to use this function.

```
<HTML>
<HEAD><TITLE>IncludeFile Test</TITLE></HEAD>
<BODY>
  <H1> Contents of 'info.txt':</H1>
  <P>
    <?--#exec cmd_argument='IncludeFile("info.txt")'-->.
  </P>
</BODY>
</HTML>
```

Contents of 'info.txt':

```
Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit...
```

When viewed in a browser, the resulting page should look somewhat as follows:




See also...

- "Include File" on page 40

### 7.3.8 SaveDataToFile()

This command stores data from an HTML form as a file in the file system. Content from the different form objects are separated by a blank line (2*CRLF).

**Note:** This command cannot be used in e-mail messages.

Syntax:

```
<?--#exec cmd_argument='SaveDataToFile("filename" [, "source"],
                                    Overwrite|Append)'-->
```

| | |
|---|---|
| filename- | Destination file. If the specified file does not exist, it will be created (provided that the path is valid). |
| source - | Optional; by specifying a form object, only data from that particular form object will be stored. Default behavior is to store data from all form objects except the ones where the name starts with underscore ('_'). |
| Overwrite\|Append- | Specifies whether to overwrite or append data to existing files. |

Default Output:

| Scenario | Default Output |
|---|---|
| Success | "Configuration stored to "*filename*" " |
| Authentication Error | "Authentication error" |
| File write error | "Could not store configuration to *"filename"* " |

Example:

The following example demonstrates how to use this function. The resulting page sends a form to itself, which is then evaluated by the SaveDataToFile command.

```
<HTML>
<HEAD><TITLE>SaveDataToFile Test</TITLE></HEAD>
<BODY>

<?--#exec cmd_argument='SaveDataToFile("\stuff.txt", "Meat", Overwrite)'-->

<FORM action="test.shtm">
  <P>
    <LABEL for="Fruit">Fruit: </LABEL><BR>
    <INPUT type="text" name="Fruit"><BR><BR>

    <LABEL for="Meat">Meat: </LABEL><BR>
    <INPUT type="text" name="Meat"><BR><BR>

    <LABEL for="Bread">Bread: </LABEL><BR>
    <INPUT type="text" name="Bread"><BR><BR>

    <INPUT type="submit" name="_submit">
  </P>
</FORM>

</BODY>
</HTML>
```

The resulting file ('\stuff.txt') will contain the value specified for the form object called 'Meat'.

**Note:** In order for this example to work, the HTML file must be named 'test.shtm'.

### 7.3.9 printf()

This function returns a formatted string which may contain data from the Anybus CompactCom module and/or application. The formatting syntax used is similar to that of the standard C-function printf().

The function accepts a template string containing zero or more formatting tags, followed by a number of arguments. Each formatting tag corresponds to a single argument, and determines how that argument shall be converted to human readable form.

Syntax:

```
<?--#exec cmd_argument='printf("template" [, argument1, ..., argumentN])'-->
```

template- Template which determines how the arguments shall be represented. May contain any number of formatting tags which are substituted by subsequent arguments and formatted as requested. The number of format tags must match the number of arguments; if not, the result is undefined.

Formatting tags are written as follows:

```
%[Flags][Width][.Precision][Modifier]type
```

See also...

- "Formatting Tags" on page 50

argument- Source arguments; optional parameters which specify the actual source of the data that shall be inserted in the template string. The number of arguments must match the number of formatting tags; if not, the result is undefined.

At the time of writing, the only allowed argument is ABCCMessage().

See also...

- "ABCCMessage()" on page 53

Default Output:

| Scenario | Default Output |
|---|---|
| Success | (printf() result) |
| ABCCMessage error | ABCCMessage error string (7-56 "Errors") |

Example:

See also...

- "ABCCMessage()" on page 53
- "Example (Get_Attribute):" on page 55

## Formatting Tags

- **Type (Required)**

  The Type-character is required and determines the basic representation as follows:

| Type Character | Representation | Example |
|---|---|---|
| c | Single character | b |
| d, i | Signed decimal integer. | 565 |
| e, E | Floating-point number in exponential notation. | 5.6538e2 |
| f | Floating-point number in normal, fixed-point notation. | 565.38 |
| g, G | %e or %E is used if the exponent is less than -4 or greater than or equal to the precision; otherwise %f is used. Trailing zeroes/decimal point are not printed. | 565.38 |
| o | Unsigned octal notation | 1065 |
| s | String of characters | Text |
| u | Unsigned decimal integer | 4242 |
| x, X | Hexadecimal integer | 4e7f |
| % | Literal %; no assignment is made | % |

- **Flags (Optional)**

| Flag Character | Meaning |
|---|---|
| - | Left-justify the result within the give width (default is right justification) |
| + | Always include a '+' or '-' to indicate whether the number is positive or negative |
| (space) | If the number does not start with a '+' or '-', prefix it with a space character instead. |
| 0 (zero) | Pad the field with zeroes instead of spaces |
| # | For %e, %E, and %f, forces the number to include a decimal point, even if no digits follow. For %x and %X, prefixes 0x or 0X, respectively. |

- **Width (Optional)**

| Width | Meaning |
|---|---|
| number | Specifies the minimum number of characters to be printed.<br>If the value to be printed is shorter than this number, the result is padded to make up the field width. The result is never truncated even if the result is larger. |
| * | The width is not specified in the format string, it is specified by an integer value preceding the argument that has to be formatted. |

- **Precision (Optional)**

  The exact meaning of this field depends on the type character:

| Type Character | Meaning |
|---|---|
| d, i, o, u, x, X | Specifies the minimum no. of decimal digits to be printed. If the value to be printed is shorter than this number, the result is padded with space. Note that the result is never truncated, even if the result is larger. |
| e, E, f | Specifies the no. of digits to be printed after the decimal point (default is 6). |
| g, G | Specifies the max. no. of significant numbers to be printed. |
| s | Specifies the max. no. of characters to be printed |
| c | (no effect) |

- **Modifier**

| Modifier Character | Meaning |
|---|---|
| hh | Argument is interpreted as SINT8 or UINT8 |
| h | Argument is interpreted as SINT16 or UINT16 |
| L | Argument is interpreted as SINT32 or UINT32 |

### 7.3.10 scanf()

This function is very similar to the printf() function described earlier, except that it is used for input rather than output. The function reads a string passed from an HTML form object, parses the string as specified by a template string, and sends the resulting data to the specified argument. The formatting syntax used is similar to that of the standard C-function scanf().

The function accepts a source, a template string containing zero or more formatting tags, followed by a number of arguments. Each argument corresponds to a formatting tag, which determines how the data read from the HTML form shall be interpreted prior sending it to the destination argument.

**Note:** This command cannot be used in e-mail messages.

Syntax:

```
<?--#exec cmd_argument='scanf("source", "template" [,
                                argument1, ..., argumentN])'-->
```

source - Name of the HTML form object from which the string shall be extracted.

template- Template which specifies how to parse and interpret the data. May contain any number of formatting tags which determine the conversion prior to sending the data to subsequent arguments. The number of formatting tags must match the number of arguments; if not, the result is undefined.

Formatting tags are written as follows:

```
%[*][Width][Modifier]type
```

See also...

- "Formatting Tags" on page 52

argument- Destination argument(s) specifying where to send the interpreted data. The number of arguments must match the number of formatting tags; if not, the result is undefined.

At the time of writing, the only allowed argument is ABCCMessage().

See also...

- "ABCCMessage()" on page 53

Default Output:

| Scenario | Default Output |
|---|---|
| Success | "Success" |
| Parsing error | "Incorrect data format" |
| Too much data for argument | "Too much data" |
| ABCC Message error | ABCCMessage error string ( "Errors" on page 56) |

Example:

See also...

- "ABCCMessage()" on page 53
- "Example (Set_Attribute):" on page 55

## Formatting Tags

- **Type (Required)**

  The Type-character is required and determines the basic representation as follows:

| Type | Input | Argument Data Type |
|---|---|---|
| c | Single character | CHAR |
| d | Accepts a signed decimal integer | SINT8<br>SINT16<br>SINT32 |
| i | Accepts a signed or unsigned decimal integer. May be given as decimal, hexadecimal or octal, determined by the initial characters of the input data:<br>Initial Characters:Format:<br>0x Hexadecimal<br>0 Octal<br>1... 9 Decimal | SINT8/UINT8<br>SINT16/UINT16<br>SINT32/UINT32 |
| u | Accepts an unsigned decimal integer. | UINT8<br>UINT16<br>UINT32 |
| o | Accepts an optionally signed octal integer. | SINT8/UINT8<br>SINT16/UINT16<br>SINT32/UINT32 |
| x, X | Accepts an optionally signed hexadecimal integer. | SINT8/UINT8<br>SINT16/UINT16<br>SINT32/UINT32 |
| e, E,<br>f,<br>g, G | Accepts an optionally signed floating point number. The input format for floating-point numbers is a string of digits, with some optional characteristics:<br>- It can be a signed value<br>- It can be an exponential value, containing a decimal rational number followed by an exponent field, which consists of an 'E' or an 'e' followed by an integer. | FLOAT |
| n | Consumes no input; the corresponding argument is an integer into which scanf writes the number of characters read from the object input. | SINT8/UINT8<br>SINT16/UINT16<br>SINT32/UINT32 |
| s | Accepts a sequence of nonwhitespace characters | STRING |
| [scanset] | Accepts a sequence of nonwhitespace characters from a set of expected bytes specified by the scanlist (e.g '[0123456789ABCDEF]')<br>A literal ']' character can be specified as the first character of the set. A caret character ('^') immediately following the initial '[' inverts the scanlist, i.e. allows all characters except the ones that are listed. | STRING |
| % | Accepts a single '%' input at this point; no assignment or conversion is done. The complete conversion specification should be '%%'. | - |

- *** (Optional)**

  Data is read but ignored. It is not assigned to the corresponding argument.

- **Width (Optional)**

  Specifies the maximum number of characters to be read.

- **Modifier (Optional)**

  Specifies a different data size.

| Modifier | Meaning |
|---|---|
| h | SINT8, SINT16, UINT8 or UINT16 |
| l | SINT32 or UINT32 |

# 7.4 Argument Functions

## 7.4.1 General Information

Argument functions are supplied as parameters to certain command functions.

General Syntax:

(Syntax depends on context)

Argument Functions:

| Function | Description | Page |
|---|---|---|
| ABCCMessage() | - | 53 |

## 7.4.2 ABCCMessage()

This function issues an object request towards an object in the module or in the host application.

Syntax:

```
ABCCMessage(object, instance, command, ce0, ce1,
            msgdata, c_type, r_type)
```

object - Specifies the Destination Object

instance- Specifies the Destination Instance

command- Specifies the Command Number

ce0 - Specifies CmdExt[0] for the command message

ce1 - Specifies CmdExt[1] for the command message

msgdata- Specifies the actual contents of the MsgData[] subfield in the command

- Data can be supplied in direct form (format depends on c_type)
- The keyword "ARG" is used when data is supplied by the parent command (e.g. scanf()).

c_type - Specifies the data type in the command (msgdata)

See also...

- "Command Data Types (c_type)" on page 54

r_type - Specifies the data type in the response (msgdata)

See also...

- "Response Data Types (r_type)" on page 54

Numeric input can be supplied in the following formats:

Decimal (e.g. 50)- (no prefix)
Octal (e.g. 043)- Prefix 0 (zero)
Hex (e.g. 0x1f)- Prefix 0x

See also...

- "Example (Get_Attribute):" on page 55
- "Example (Set_Attribute):" on page 55

- **Command Data Types (c_type)**

  For types which support arrays, the number of elements can be specified using the suffix '[n]', where 'n' specifies the number of elements. Each data element must be separated by space.

| Type | Supports Arrays | Data format (as supplied in msgdata) |
|---|---|---|
| BOOL | Yes | 1 |
| SINT8 | Yes | -25 |
| SINT16 | Yes | 2345 |
| SINT32 | Yes | -2569 |
| UINT8 | Yes | 245 |
| UINT16 | Yes | 40000 |
| UINT32 | Yes | 32 |
| CHAR | Yes | A |
| STRING | No | "abcde"<br>**Note:** Quotes can be included in the string if preceded by backslash('\')<br>Example: *"We usually refer to it as \"the Egg\" "* |
| FLOAT | Yes | 5.6538e2 |
| NONE | No | Command holds no data, hence no data type |

- **Response Data Types (r_type)**

  For types which support arrays, the number of elements can be specified using the suffix '[n]', where 'n' specifies the number of elements.

| Type | Supports Arrays | Comments |
|---|---|---|
| BOOL | Yes | Optionally, it is possible to exchange the BOOL data with a message based on the value (true or false). In such case, the actual data type returned from the function will be STRING.<br>Syntax: BOOL<true><false><br>For arrays, the format will be BOOL[n]<true><false>. |
| SINT8 | Yes | - |
| SINT16 | Yes | - |
| SINT32 | Yes | - |
| UINT8 | Yes | This type can also be used when reading ENUM data types from an object. In such case, the actual ENUM value will be returned. |
| UINT16 | Yes | - |
| UINT32 | Yes | - |
| CHAR | Yes | - |
| STRING | No | - |
| ENUM | No | When using this data type, the ABCCMessage() function will first read the ENUM value. It will then issue a 'Get Enum String'-command to retrieve the actual enumeration string. The actual data type in the response will be STRING. |
| FLOAT | Yes | - |
| NONE | No | Response holds no data, hence no data type |

**IMPORTANT:** *It is important to note that the message will be passed transparently to the addressed object. The SSI engine performs no checks for violations of the object addressing scheme, e.g. a malformed Get_Attribute request which (wrongfully) includes message data will be passed unmodified to the object, even though this is obviously wrong. Failure to observe this may cause loss of data or other undesired side effects.*

Example (Get_Attribute):

This example shows how to retrieve the IP address using printf() and ABCCMessage().

```
<?--#exec cmd_argument='printf( "%u.%u.%u.%u",
                                ABCCMessage(4,3,1,5,0,0,NONE,UINT8[4] ) )'-->
```

| Variable | Value | Comments |
|---|---|---|
| object | 4 | Network Configuration Object (04h) |
| instance | 3 | Instance #3 (IP address) |
| command | 1 | Get_attribute |
| ce0 | 5 | Attribute #5 |
| ce1 | 0 | - |
| msgdata | 0 | - |
| c_type | NONE | Command message holds no data |
| r_type | UINT8[4] | Array of 4 unsigned 8-bit integers |

See also...

- 7-49 “printf()”

Example (Set_Attribute):

This example shows how to set the IP address using scanf() and ABCCMessage(). Note the special parameter value ‘ARG’, which instructs the module to use the passed form data (parsed by scanf() ).

```
<?--#exec cmd_argument='scanf("IP", "%u.%u.%u.%u",
                              ABCCMessage(4,3,2,5,0,ARG,UINT8[4],NONE ) )'-->
```

| Variable | Value | Comments |
|---|---|---|
| object | 4 | Network Configuration Object (04h) |
| instance | 3 | Instance #3 (IP address) |
| command | 2 | Set_attribute |
| ce0 | 5 | Attribute #5 |
| ce1 | 0 | - |
| msgdata | ARG | Use data parsed by scanf() call |
| c_type | UINT8[4] | Array of 4 unsigned 8-bit integers |
| r_type | NONE | Response message holds no data |

See also...

- “scanf()” on page 51

### Errors

In case an object request results in an error, the error code in the response will be evaluated and translated to human readable form as follows:

| Error Code | Output |
|---|---|
| 0 | “Unknown error” |
| 1 | “Unknown error” |
| 2 | “Invalid message format” |
| 3 | “Unsupported object” |
| 4 | “Unsupported instance” |
| 5 | “Unsupported command” |
| 6 | “Invalid CmdExt[0]” |
| 7 | “Invalid CmdExt[1]” |
| 8 | “Attribute access is not set-able” |
| 9 | “Attribute access is not get-able” |
| 10 | “Too much data in msg data field” |
| 11 | “Not enough data in msg data field” |
| 12 | “Out of range” |
| 13 | “Invalid state” |
| 14 | “Out of resources” |
| 15 | “Segmentation failure” |
| 16 | “Segmentation buffer overflow” |
| 17... 255 | “Unknown error” |

See also...


- “SSI Output Configuration” on page 57

# 7.5 SSI Output Configuration

Optionally, the SSI output can be permanently changed by adding the file '\output.cfg'.

File format:

```
[ABCCMessage_X]
0:"Success string"
1:"Error string 1"
2:"Error string 2"
...
16:"Error string 16"

[GetConfigItem_X]
0:"Success string"
1:"Authentication error string"
2:"File open error string"
3:"Tag not found string"

[SetConfigItem_X]
0:"Success string"
1:"Authentication error string"
2:"File open error string"
3:"File write error string"

[IncludeFile_X]
0:"Success string"
1:"Authentication error string"
2:"File readS error string"

[scanf_X]
0:"Success string"
1:"Parsing error string"

[ChangeLanguage_X]
0:"Success string"
1:"Change error string"
```

Each error code corresponds to a dedicated output string, labelled from 1 to 16.

See also...

- "Errors" on page 56

Use "%s" to include the name of the file.

Use "%s" to include the name of the file.

Use "%s" to include the name of the file.

All content above can be included in the file multiple times changing the value 'X' in each tag for different languages. The module will then select the correct output string based on the language settings. If no information for the selected language is found, it will use the default SSI output.

| Value of X | Language |
|---|---|
| 0 | English |
| 1 | German |
| 2 | Spanish |
| 3 | Italian |
| 4 | French |

See also...

- "SsiOutput()" on page 45

# 8. JSON

## 8.1 General Information

JSON is an acronym for JavaScript Object Notation and an open standard format for storing and exchanging data in an organized and intuitive way. It is used as an alternative to XML, to transmit data objects consisting of attribute - value pairs between a server and a web application.

### Access

The JSON resources should be password protected. Add password protection by adding a file called web_accs.cfg in the root directory. See "Authorization" on page 38 for more information.

## 8.2 JSON Objects

### 8.2.1 ADI

#### info.json

GET adi/info.json[?callback=<function>].

This object holds data common to all ADIs that are static during runtime. Optionally, a callback may be passed to the GET-request for JSONP output.

| Name | Data Type | Note |
|---|---|---|
| dataformat | Number | 0 = Little endian<br>1 = Big endian<br>(Affects value, min and max representations) |
| numadis | Number | Total number of ADIs |
| webversion | Number | Web/JSON API version |

JSON object layout:

```
{
   "dataformat": 0,
   "numadis":    123,
   "webversion": 1
}
```

### data.json

GET adi/data.json?offset=<offset>&count=<count>[&callback=<function>].

This object call fetches values for up to <count> ADIs, starting from <offset> in a list sorted by ADI order number. The values may change at any time during runtime. Optionally, a callback may be passed to the GET-request for JSONP output.

JSON object layout:

```
[
   "FF",
   "A201",
   "01FAC105"
]
```

### metadata.json

GET adi/metadata.json?offset=<offset>&count=<count>[&callback=<function>].

This object call fetches metadata for up to <count> ADIs, starting from <offset> in a list sorted by ADI order number. This data is static during runtime. Optionally, a callback may be passed to the GET-request for JSONP output.

| Name | Data Type | Note |
|---|---|---|
| instance | Number | - |
| name | String | May be NULL if no name is present. |
| numelements | Number | - |
| datatype | Number | - |
| min | String | Minimum value. May be NULL if no minimum value is present. |
| max | String | Maximum value. May be NULL of no maximum value is present. |
| access | Number | Bit 0: Read access<br>Bit 1: Write access |

JSON object layout:

```
[
   {
      "instance":    1,
      "name":        "Temperature threshold",
      "numelements": 1,
      "datatype":    0,
      "min":         "00",
      "max":         "FF",
      "access":      0x03
   }
   {
      nine more...
   }
]
```

### enum.json

GET adi/enum.json?inst=<instance>[&value=<element>][&callback=<function>].

This object call fetches enum strings for the instance <instance>. If an <element> is specified, only the enum string for that value is returned. If no enum strings are available, an empty list is returned. Optionally, a callback may be passed to the GET-request for JSONP output.

| Name | Data Type | Note |
|---|---|---|
| string | String | - |
| value | Number | - |

JSON object layout:

```
[
   {
      "string": "String for value 1",
      "value": 1
   },
   {
      "string": "String for value 2",
      "value": 2
   },
   ...
]
```

### update.json

POST adi/update.json - form data:
inst=<instance>&value=<data>[&elem=<element>][&callback=<function>].

Updates the value of an ADI for the specified ADI instance <instance>. The value, <data>, shall be hex formatted (see "Hex Format Explained" on page 66 for more information). If <element> is specified, only the value of the specified element is updated. In this case, <data> shall only update that single element value. When <element> is not specified, <data> shall represent the entire array value. Optionally, a callback may be passed to the request for JSONP output.

| Name | Data Type | Note |
|---|---|---|
| result | Number | 0 = success |

POST adi/update.json - form data: inst=15&value=FF01

```
{
   "result" : 0
}
```

## 8.2.2 Module

### info.json

GET module/info.json.

| Name | Data Type | Note |
|---|---|---|
| modulename | String | - |
| serial | String | 32 bit hex ASCII |
| fwver | Array of Number | (major, minor, build) |
| uptime | Array of Number | [high, low] milliseconds (ms) |
| cpuload | Number | CPU load in % |

JSON object layout:

```
{
   "modulename":  "ABCC M40",
   "serial":      "ABCDEF00",
   "fwver":       [ 1, 5, 0 ],
   "uptime":      [ 5, 123456 ],
   "cpuload":     55
}
```

## 8.2.3 Network

### ethstatus.json

GET network/ethstatus.json.

| Name | Data Type | Note |
|---|---|---|
| mac | String | 6 byte hex |
| comm1 | Object | See object definition in the table below |
| comm2 | Object | See object definition in the table below |

**Comm Object Definition:**

| Name | Data Type | Note |
|---|---|---|
| link | Number | 0: No link<br>1: Link |
| speed | Number | 0: 10 Mbit<br>1: 100 Mbit |
| duplex | Number | 0: Half<br>1: Full |

JSON object layout:

```
{
   "mac":          "003011FF0201",
   "comm1":        {
       "link":         1,
       "speed":        1,
       "duplex":       1
   }
   "comm2":        {
       "link":         0,
       "speed":        0,
       "duplex":       0
   }
}
```

### ipstatus.json & ipconf.json

These two object share the same data format. The object ipconf.json returns the configured IP settings, and ipstatus.json returns the actual values that are currently used. ipconf.json can also be used to alter the IP settings.

GET network/ipstatus.json, or GET network/ipconf.json.

| Name | Data Type | Note |
|---|---|---|
| dhcp | Number | - |
| addr | String | - |
| subnet | String | - |
| gateway | String | - |
| dns1 | String | - |
| dns2 | String | - |
| hostname | String | - |
| domainname | String | - |

```
{
   "dhcp":       0,
   "addr":       "192.168.0.55",
   "subnet":     "255.255.255.0",
   "gateway":    "192.168.0.1",
   "dns1":       "10.10.55.1",
   "dns2":       "10.10.55.2"
   "hostname":   "<hostname>",
   "domainname": "hms.se"
}
```

To change IP settings, use network/ipconf.json. It accepts any number of arguments from the list above. Values should be in the same format.

Example:

GET ipconf.json?dhcp=0&addr=10.11.32.2&hostname=abcc123&domainname=hms.se

### ethconf.json

GET network/ethconf.json.

| Name | Data Type | Note |
|---|---|---|
| comm1 | Number | - |
| comm2 | Number | - |

### ifcounters.json

GET network/ifcounters.json?port=<port>. The argument <port> is either 1 or 2.

| Name | Data Type | Note |
|---|---|---|
| inoctets | Number | IN: bytes |
| inucast | Number | IN: unicast packets |
| innucast | Number | IN: broadcast and multicast packets |
| indiscards | Number | IN: discarded packets |
| inerrors | Number | IN: errors |
| inunknown | Number | IN: unsupported protocol type |
| outoctets | Number | OUT: bytes |
| outucast | Number | OUT: unicast packets |
| outnucast | Number | OUT: broadcast and multicast packets |
| outdiscards | Number | OUT: discarded packets |
| outerrors | Number | OUT: errors |

### mediacounters.json

GET network/mediacounters.json?port=<port>. The argument <port> is either 1 or 2.

| Name | Data Type | Note |
|---|---|---|
| align | Number | Frames received that are not an integral number of octets in length |
| fcs | Number | Frames received that do not pass the FCS check |
| singlecoll | Number | Successfully transmitted frames which experienced exactly one collision |
| multicoll | Number | Successfully transmitted frames which experienced more than one collision |
| latecoll | Number | Number of collisions detected later than 512 bit times into the transmission of a packet |
| excesscoll | Number | Frames for which transmissions fail due to excessive collisions |
| sqetest | Number | Number of times SQE test error is generated |
| deferredtrans | Number | Frames for which the first transmission attempt is delayed because the medium is busy |
| macrecerr | Number | Frames for which reception fails due to an internal MAC sublayer receive error |
| mactranserr | Number | Frames for which transmission fails due to an internal MAC sublayer transmit error |
| cserr | Number | Times that the carrier sense was lost or never asserted when attempting to transmit a frame |
| toolong | Number | Frames received that exceed the maximum permitted frame size |
| tooshort | Number | Frames received that are shorter than the lowest permitted frame size |

### nwstats.json

GET network/nwstats.json.

This object lists available statistics data. The data available depends on the product.

Example output:

```
[]
or
[ { "identifier": "eip", "title": "EtherNet/IP Statistics" } ]
or
[
   { "identifier": "bacnet",    "title": "BACnet/IP Statistics" },
   { "identifier": "bacnetae",  "title": "BACnet Alarm and Event" },
   { "identifier": "bacnetapl", "title": "BACnet APL Statistics" }
]
```

Get network specific statistics:

GET network/nwstats.json?get=<ID>. <ID> is an “identifier” value returned from the previous command (“eip”, for example)

```
[
   { "name": "Established Class1 Connections", "value": 0 },
   { "name": "Established Class3 Connections", "value": 1 }
]
```

## 8.2.4 Services

### smtp.json

GET services/smtp.json.

Note: Password is not returned when retrieving the settings.

| Name | Data Type | Note |
|---|---|---|
| server | String | - |
| user | String | - |

## 8.2.5 Hex Format Explained

The metadata max and min fields and the ADI values are ABP data encoded in a hex format. If the data type is an integer, the endianness used is determined by the data format field found in adi/info.json (see “info.json” on page 58).

Examples:
The value “5” encoded as a UINT16, with data format = 0 (little endian):

```
0500
```

The character array “ABC” encoded as CHAR[3] (data format is not relevant for CHAR):

```
414243
```

# 9. SNMP Agent

## 9.1 General

Simple Network Management Protocol (SNMP, see RFC1157 standard) is used in network management systems to monitor network-attached devices for conditions that warrant administrative attention. A management agent is installed in the managing station, and exchanges data via get and set requests.

## 9.2 Management Information (MIB)

A MIB is a device database that is accessed by an SNMP agent. The Anybus CompactCom 40 PROFINET IRT 2-Port module supports standardized MIBs: LLDP-MIB and MIB-II. Standardized MIBs are defined in RFC standards and contain variables that are divided into so called groups. The host application can change the values of some of the variables for the MIB-II.

## 9.3 MIB-II

The MIB-II of the Anybus CompactCom 40 PROFINET IRT module contains the system- and interfaces group. The following tables show the variables according to the MIB-II standard (RFC1213) for monitoring the device status. The access authorizations refer to access via the SNMP protocol.

### 9.3.1 System Group Variables

| Variable | Access Authorizations | Description | Source of Origin |
|---|---|---|---|
| sysDescr | Read only | Description of the device. Data type: Display-String(only printable ASCII characters). Max 255 characters. Factory default setting: “HMS Industrial Networks Anybus-CompactCom 40” | PROFINET Object; Instance attribute 19 - System Description See “Instance Attributes (Instance #1)” on page 131 |
| sysObjectID | Read only | N/A. Value=0 | Internal |
| sysUpTime | Read only | Time since last power up (in hundredths of a second) | Internal |
| sysContact | Read/Write | Identification of the contact person for the device, including contact information. Data type: Displaystring. Max 255 characters. Factory default setting: empty string (“”) | Internal |
| sysName | Read/Write | Name of the device. Data type: Displaystring. Max 255 characters. Factory default setting: empty string (“”) | Internal |
| sysLocation | Read/Write | Physical location of the device (IM Tag Location). Data type: (DisplayString). Max 255 characters. Factory default setting: empty string (“”) | Internal |
| sysServices | Read only | Functionality of the device. Value=74, which indicates that the device has functionality that represents layers 2(switch), 4(TCP) and 7(Application) in the OSI model. | Internal |

### 9.3.2 Interfaces Group Variables

Access authorizations for all variables are read only with values from internal sources. The number in brackets refers to the port number (1 - Port 1, 2 - Port 2, 3 - Internal port)

| Variable | Data Type | Value[a] | Description |
|---|---|---|---|
| ifNumber | integer | 3 | Number of network interfaces present. Constant |
| IfIndex(1..3) | integer | ifIndex(1) = 1<br>ifIndex(2) = 2<br>ifIndex(3) = 3 | Unique value for each interface. Constant |
| ifDescr(1..3) | octetstring | ifDescr(1) = “port-001”<br>ifDescr(2) = “port-002”<br>ifDescr(3) = “port-internal” | Information about the interface. ifDescr(1) must equal “port-001”and ifDescr(2) = “port-002” to be compatible with the STEP7 topology scanner. |
| ifType(1..3) | integer | 6 (“Ethernet-csmacd”) | Type of interface |
| ifMtu(1..3) | integer | 1500 | Size of largest datagram that can be sent/received on the interface, specified in octets |
| ifSpeed(1..3) | gauge | 0 or 100 000 000 | Data transfer rate of the Ethernet port in bits per second. The speed is only shown for ports where the link status is “up”. |
| ifPhysAddress(1..3) | octetstring | | MAC address for the ports |
| ifAdminStatus(1..3) | integer | 1 (“up”) | Desired state of the Ethernet port |
| ifOperStatus(1..3) | integer | 1 (“up”) or 2 (“down”) | Current operating state of the Ethernet port. (Link = “up”, No link = “down”. |
| ifLastChange(1..3) | timeticks | Time when state changed, except ifLast-Change(3) = 0 | Time (since start-up) when the port changed to its current state, see previous variable. Indicated in multiples of hundredths of a second |
| ifInOctets(1..3) | counter | ifInOctets(1..3) = Number of octets | Total number of octets received on the interface, including framing characters |

| Variable | Data Type | Value[a] | Description |
|---|---|---|---|
| ifInUcastPkts(1..3) | counter | ifInUcastPkts(1..3) = Number of unicast packets | Number of subnetwork-unicast packets delivered to a higher-layer protocol |
| ifInNUcastPkts(1..3) | counter | ifInNUcastPkts(1..3) = Number of non-unicast packets | Number of non-unicast (i.e. subnetwork-broadcast or subnetwork-multicast) delivered to a higher-layer protocol. |
| ifInDiscards(1..3) | counter | ifInDiscards(1..3) = number of discarded packets | Number of inbound packets which were discarded, without any error detected, not to be delivered to a higher-layer protocol. (One reason to discard packages might be to free up buffer space) |
| ifInErrors(1..3) | counter | ifInErrors(1..3) = number of error packets | Number of inbound packets with errors |
| ifInUnknownProtos(1..3) | counter | ifInUnknownProtos(1..3) = Number of unknown packets | Number of packets received via the interface, discarded because of an unknown or unsupported protocol. |
| ifOutOctets(1..3) | counter | ifOutOctets(1..3) = Number of octets | Total number of octets transmitted out from the interface, including framing characters |
| ifOutUcastPkts(1..3) | counter | ifOutUcastPkts(1..3) = Number of unicast packets | Total number of packets that higher-level protocols requested to be transmitted to a subnetwork-unicast address, including those that were discarded or not sent. |
| ifOutNUcastPkts(1..3) | counter | ifOutNUcastPkts(1..3) = Number of non-unicast packets | Total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e. a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent. |
| ifOutDiscards(1..3) | counter | ifOutDiscards(1..3) = Number of discarded packets | Number of outbound packets which were discarded without any error detected, not to be transmitted. (One reason to discard packages might be to free up buffer space) |
| ifOutErrors(1..3) | counter | ifOutErrors(1..3) = Number of error packets | Number of outbound packets that could not be transmitted due to errors |
| ifOutQLen(1..3) | gauge | ifOutQLen(1..3) = Number of packets in queue | Length of the output packet queue (in packets), |
| ifSpecific(1..3) | objid | .0.0 | Reference to MIB definitions specific to the particular media being used to realize the interface. Here no reference is available, so a fixed value is used for all ports. |

a. If nothing else is specified, the value of a variable is 0

# 10. Media Redundancy Protocol (MRP)

## 10.1 General

Media Redundancy Protocol (MRP) is a PROFINET specific ring protocol ensuring redundancy in the network, which can significantly decrease network downtime. It is a token based ring protocol with a master-slave hierarchy.

All the nodes in the PROFINET network part of the ring are connected using ring topology (that is, the last node is connected directly to the first node). If, at any point, the connection between two nodes would break, the data will flow the other way instead, guaranteeing that data can be sent to/from the IO Controller to the IO Device(s). The self-healing time is approximately 200 ms.

The Media Redundancy Master (MRM) is responsible for checking the functional capability of the ring network, by sending out cyclic tokens. The Media Redundancy Clients (MRC) basically work as switches that pass on the tokens. The Anybus CompactCom module supports acting as a Media Redundancy Client (MRC). It also supports propagating link change to the Media Redundancy Master.

## 10.2 GSDML Entries

MRP functionality is enabled by default in the GSD file. The settings for MRP is located at the Device Access Point (DAP). Within the <InterfaceSubmoduleItem …> the role the Anybus module can play for MRP is defined. This shall be set to “Client” with the keyword <MediaRedundancy SupportedRole="Client"/>.

For each physical port there are two keywords in the <PortSubmoduleItem …> section <PortSubmoduleItem … SupportsRingportConfig="true" IsDefaultRingport="true" … />. These are set to “true” by default. To disable MRP, these two shall be set to “false”.

# 11. Anybus Module Objects

## 11.1 General Information

This chapter specifies the Anybus Module Object implementation in the module.

Standard Objects:



Network Specific Objects:

# 11.2 Anybus Object (01h)

## Category

Basic

## Object Description

This object assembles all common Anybus data, and is described thoroughly in the general Anybus CompactCom 40 Software Design Guide.

## Supported Commands

| | |
|---|---|
| Object: | Get_Attribute |
| Instance: | Get_Attribute<br>Set_Attribute<br>Get_Enum_String |

## Object Attributes (Instance #0)

(Consult the general Anybus CompactCom 40 Software Design Guide for further information.)

## Instance Attributes (Instance #1)

### Basic

| # | Name | Access | Type | Value |
|---|---|---|---|---|
| 1 | Module type | Get | UINT16 | 0403h (Standard Anybus CompactCom 40) |
| 2... 11 | - | - | - | Consult the general Anybus CompactCom 40 Software Design Guide for further information. |
| 12 | LED colors | Get | struct of:<br>UINT8(LED1A)<br>UINT8(LED1B)<br>UINT8(LED2A)<br>UINT8(LED2B) | Value:Color:<br>01h Green<br>02h Red<br>01h Green<br>02h Red |
| 13... 16 | - | - | - | Consult the general Anybus CompactCom 40 Software Design Guide for further information. |
| 17 | Virtual attributes | Get/Set | | |
| 18 | Black list/White list | Get/Set | | |
| 19 | Network time | Get | UINT64 | 0 (Not supported) |

# 11.3 Diagnostic Object (02h)

## Category

Extended, advanced

## Object Description

This object provides a standardized way of handling host application events & diagnostics, and is thoroughly described in the general Anybus CompactCom 40 Software Design Guide.

## Supported Commands

| | |
|---|---|
| Object: | Get_Attribute<br>Create<br>Delete |
| Instance: | Get_Attribute |

## Object Attributes (Instance #0)

| # | Name | Access | Data Type | Value |
|---|---|---|---|---|
| 1... 4 | - | - | - | Consult the general Anybus CompactCom 40 Software Design Guide for further information. |
| 11 | Max no. of instances | Get | UINT16 | 5+1[a] |
| 12 | - | - | - | Consult the general Anybus CompactCom 40 Software Design Guide for further information. |

a. Of the maximum number of instances there should always be one instance reserved for an event of severity level ‘Major, unrecoverable’, to force the module into the ‘EXCEPTION’-state.

## Instance Attributes

### Extended

| # | Name | Access | Type | Value |
|---|---|---|---|---|
| 1 | Severity | Get | UINT8 | Consult the general Anybus CompactCom 40 Software Design Guide for further information. |
| 2 | Event Code | Get | UINT8 | |

### Advanced

| # | Name | Access | Type | Value |
|---|---|---|---|---|
| 3 | NW specific data | Get | Array of UINT8 | Optional network specific information.<br>See also...<br>- "Details: Network Specific Data" on page 75 |

Major unrecoverable events cause the module to disconnect itself from the network, thus preventing network participation. Other severity levels either produce a Channel Diagnostic entry/alarm or a Generic Diagnostic entry/alarm, depending on the Event Code:

| Event Code | Result |
|---|---|
| 0...FEh | Module issues a Channel Diagnostic entry/alarm. The Event Code will be translated and represented as the Channel Error Type as follows:<br>Code: / Event (Anybus): / Channel Error Type (PROFINET):<br>10h Generic Error — Error<br>20h Current — Short circuit<br>21h Current, device input side — Short circuit<br>22h Current, inside the device — Short circuit<br>23h Current, device output side — Short circuit<br>30h Voltage — Overvoltage<br>31h Mains Voltage — Overvoltage<br>32h Voltage inside the device — Overvoltage<br>33h Output Voltage — Overvoltage<br>40h Temperature — Overtemperature<br>41h Ambient Temperature — Overtemperature<br>42h Device Temperature — Overtemperature<br>50h Device Hardware — Error<br>60h Device Software — Error<br>61h Internal Software — Error<br>62h User Software — Error<br>63h Data Set — Error<br>70h Additional Modules — Error<br>80h Monitoring — Error<br>81h Communication — Error<br>82h Protocol Error — Error<br>90h External Error — Error<br>F0h Additional Functions — Error |
| FFh | Module issues a Generic Diagnostic entry/alarm based on network specific data.<br>See also...<br>- "Details: Network Specific Data" on page 75 |

## Details: Network Specific Data

Network specific diagnostic data serves as the payload in the PROFINET diagnostic alarm. The data contains an identifier (UserStructureIdentifier) that describes the structure of the data.

The following identifier values are supported:

- 8000h (Channel Diagnostic)
- 8002h (Extended Channel Diagnostic)
- 8003h (Qualified Channel Diagnostic)

| Byte | Contents |
|---|---|
| 1 | UserStructureIdentifier, low byte |
| 2 | UserStructureIdentifier, high byte |
| 3... 16 | Data |

## Object Error Codes

| Code | Error |
|---|---|
| 03h | API does not exist |
| 04h | No module inserted in the specified slot |
| 05h | No submodule inserted in the specified subslot |
| 06h | Slot number specified is out-of-range |
| 07h | Subslot number specified is out-of-range |
| 08h | Failed to add the channel diagnostic entry |
| 09h | Failed to send the channel diagnostic alarm |
| 0Ah | Channel number out-of-range |
| 0Bh | ChannelPropType out-of-range |
| 0Ch | ChannelPropDir out-of-range |
| 0Dh | ChannelPropAcc out-of-range |
| 0Eh | ChannelPropMaintReq out-of-range |
| 0Fh | ChannelPropMaintDem out-of-range |
| 10h | UserStructIdent out-of-range |
| 11h | ChannelErrType out-of-range |
| FFh | Unknown error |

# 11.4 Network Object (03h)

## Category

Basic

## Object Description

For more information regarding this object, consult the general Anybus CompactCom 40 Software Design Guide.

## Supported Commands

Object: Get_Attribute

Instance: Get_Attribute
Set_Attribute
Get_Enum_String
Map_ADI_Write_Area
Map_ADI_Read_Area

## Object Attributes (Instance #0)

(Consult the general Anybus CompactCom 40 Software Design Guide for further information.)

## Instance Attributes (Instance #1)

### Basic

| # | Name | Access | Type | Value |
|---|---|---|---|---|
| 1 | Network type | Get | UINT16 | 0089h |
| 2 | Network type string | Get | Array of CHAR | 'PROFINET IRT' |
| 3 | Data format | Get | ENUM | 01h (MSB first) |
| 4 | Parameter data support | Get | BOOL | True |
| 5 | Write process data size | Get | UINT16 | Current write process data size (in bytes)<br>Updated on every successful Map_ADI_Write_Area[a] |
| 6 | Read process data size | Get | UINT16 | Current read process data size (in bytes)<br>Updated on every successful Map_ADI_Read_Area[a] |
| 7 | Exception Information | Get | UINT8 | Additional information available if the module has entered the EXCEPTION state.<br>Value:Meaning:<br>01h Illegal value<br>02h Wrong data size<br>03h Illegal response |
| 8... 10 | - | - | - | Consult the general Anybus CompactCom 40 Software Design Guide for further information. |

a. Consult the general Anybus CompactCom 40 Software Design Guide for further information.

# 11.5 Network Configuration Object (04h)

## Category

Extended, advanced

## Object Description

This object holds network specific configuration parameters that may be set by the end user. A reset command (factory default) issued towards this object will result in all instances being set to their default values.

See also...

- "Communication Settings" on page 15
- "E-mail Client" on page 39

**Note:** Allowing the following instances to be set by the host application during start-up will inhibit the possibility to pass conformance tests.

## Supported Commands

| | |
|---|---|
| Object: | Get_Attribute<br>Reset |
| Instance: | Get_Attribute<br>Set_Attribute<br>Get_Enum_String |

## Object Attributes (Instance #0)

| # | Name | Access | Type | Value | Description |
|---|---|---|---|---|---|
| 3 | Number of instances | Get | UINT16 | 17 | Supported number of instances |
| 4 | Highest instance number | Get | UINT16 | 21 | Highest instance number |

(Consult the general Anybus CompactCom 40 Software Design Guide for further information.)

## Instance Attributes (Instance #3, IP Address)

Value is used after module reset.

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'IP address' |
| 2 | Data type | Get | UINT8 | 04h (= UINT8) |
| 3 | Number of elements | Get | UINT8 | 04h (four elements) |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of UINT8 | Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0) |

a. Multilingual, see "Instance Attributes (Instance #20, Station Name)" on page 82.

**Note**: This attribute should not be set by the application at every power on, as this would cause certification problems.

## Instance Attributes (Instance #4, Subnet Mask)

Value is used after module reset.

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'Subnet mask' |
| 2 | Data type | Get | UINT8 | 04h (= UINT8) |
| 3 | Number of elements | Get | UINT8 | 04h (four elements) |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of UINT8 | Valid range: 0.0.0.0. - 255.255.255.255 (Default =0.0.0.0) |

a. Multilingual, see "Multilingual Strings" on page 83.

**Note**: This attribute should not be set by the application at every power on, as this would cause certification problems.

## Instance Attributes (Instance #5, Gateway)

Value is used after module reset.

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'Gateway' |
| 2 | Data type | Get | UINT8 | 04h (= UINT8) |
| 3 | Number of elements | Get | UINT8 | 04h (four elements) |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of UINT8 | Valid range: 0.0.0.0. - 255.255.255.255 (Default =0.0.0.0) |

a. Multilingual, see "Multilingual Strings" on page 83.

**Note**: This attribute should not be set by the application at every power on, as this would cause certification problems.

## Instance Attributes (Instance #6, DHCP)

Value is used after module reset.

### Extended

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'DHCP' |
| 2 | Data type | Get | UINT8 | 08h (= ENUM) |
| 3 | Number of elements | Get | UINT8 | 01h (one element) |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value[a] | Get/Set | ENUM | Value:Enum. String:Meaning:<br>00h 'Disable' DHCP disabled (default)<br>01h 'Enable' DHCP enabled |

a. Multilingual, see "Multilingual Strings" on page 83.

**Note**: Do not set this unless the end user explicitly would like to turn DHCP on. Normally the PROFINET IO Controller assigns the IP address.

## Instance Attributes (Instance #9, DNS1)

### Advanced

This instance holds the address to the primary DNS server. Changes are valid after reset.

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'DNS1' |
| 2 | Data type | Get | UINT8 | 04h (= UINT8) |
| 3 | Number of elements | Get | UINT8 | 04h |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of UINT8 | Valid range: 0.0.0.0. - 255.255.255.255 (Default =0.0.0.0) |

a. Multilingual, see "Multilingual Strings" on page 83.

## Instance Attributes (Instance #10, DNS2)

This instance holds the address to the secondary DNS server. Changes are valid after reset.

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'DNS2' |
| 2 | Data type | Get | UINT8 | 04h (= UINT8) |
| 3 | Number of elements | Get | UINT8 | 04h |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of UINT8 | Valid range: 0.0.0.0. - 255.255.255.255 (Default =0.0.0.0) |

a. Multilingual, see "Multilingual Strings" on page 83.

## Instance Attributes (Instance #11, Host name)

This instance holds the host name of the module. Changes are valid after reset.

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'Host name' |
| 2 | Data type | Get | UINT8 | 07h (= CHAR) |
| 3 | Number of elements | Get | UINT8 | 40h |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of CHAR | Host name, 64 characters (pad with space to full length) |

a. Multilingual, see "Multilingual Strings" on page 83.

## Instance Attributes (Instance #12, Domain name)

This instance holds the domain name. Changes are valid after reset.

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'Domain name' |
| 2 | Data type | Get | UINT8 | 07h (= CHAR) |
| 3 | Number of elements | Get | UINT8 | 30h |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of CHAR | Domain name, 48 characters (pad with space to full length) |

a. Multilingual, see "Multilingual Strings" on page 83.

## Instance Attributes (Instance #13, SMTP Server)

This instance holds the SMTP server address. Changes are valid after reset.

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'SMTP Server' |
| 2 | Data type | Get | UINT8 | 07h (= CHAR) |
| 3 | Number of elements | Get | UINT8 | 40h |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of UINT8 | SMTP server address, 64 characters. Dotted decimal format or server name (pad with space to full length). |

a. Multilingual, see "Multilingual Strings" on page 83.

## Instance Attributes (Instance #14, SMTP User)

This instance holds user name for the SMTP account. Changes are valid after reset.

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'SMTP User' |
| 2 | Data type | Get | UINT8 | 07h (= CHAR) |
| 3 | Number of elements | Get | UINT8 | 40h |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of UINT8 | SMTP account user name, 64 characters<br>(pad with space to full length) |

a. Multilingual, see "Multilingual Strings" on page 83.

## Instance Attributes (Instance #15, SMTP Password)

This instance holds the password for the SMTP account. Changes are valid after reset.

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'SMTP Pswd' |
| 2 | Data type | Get | UINT8 | 07h (= CHAR) |
| 3 | Number of elements | Get | UINT8 | 40h |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of UINT8 | SMTP account password, 64 characters<br>(pad with space to full length) |

a. Multilingual, see "Multilingual Strings" on page 83.

## Instance Attributes (Instance #20, Station Name)

The Station Name identifies the Anybus module on PROFINET. If this value is changed by the host application during runtime, a reset is required in order for changes to have effect. Changes made through DCP will have immediate effect, however.

The Station Name field shall be coded as data type CHAR with 1 to 240 characters. The definition of RFC 5890 and the following syntax applies:

- 1 or more labels, separated by [.]
- Total length is 1 to 240
- Label length is 1 to 63
- Labels consist of [a-z, 0-9, -]
- Labels do not start with [-]
- Labels do not end with [-]
- The first label must not have the form “port-xyz” or “port-xyz-abcde”, where a, b, c, d, e, x, y, z = 0...9, to avoid similarity with the field AliasNameValue
- Station names must not have the form n.n.n.n, where n = 0...999

**Note**: Be sure to verify that the Station Name parameter value is correct, according to the criteria above. No verification checks will be made by the module, until after the application has issued “setup complete”. A faulty Station Name will then be discarded (set to an empty string) without any warning.

### Extended

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | ‘Station name’ |
| 2 | Data type | Get | UINT8 | 07h (= CHAR) |
| 3 | Number of elements | Get/Set | UINT8 | F0h |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | Array of CHAR | The current Station name |
| 6 | Configured value | Get | Array of CHAR | The configured value that will be used after restart |

a. Multilingual, see “Multilingual Strings” on page 83.

**Note 1**: This attribute shall normally not be set by the application. The station name is normally set by the end user via the network. The host application shall use this attribute when the end user has the possibility to edit the station name through the application, and chooses to do so.

**Note 2**: This attribute should not be set by the application at every power on, as this would cause certification problems.

## Instance Attributes (Instance #21, F-Address)

This instance holds the F-Address, which is the PROFIsafe address for the safety module. If this value is changed by the host application during runtime, a reset is required in order for changes to have effect.

### Extended

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Name[a] | Get | Array of CHAR | 'F-Address' |
| 2 | Data type | Get | UINT8 | 05h (= UINT16) |
| 3 | Number of elements | Get/Set | UINT8 | 01h |
| 4 | Descriptor | Get | UINT8 | 03h (read/write access) |
| 5 | Value | Get/Set | UINT16 | F-Address set by the host application.<br>Range: 1 - 65534 (Default: 1) |
| 6 | Configured value | Get | UINT16 | The configured value that will be used after restart |

a. Multilingual, see "Multilingual Strings" on page 83.

## Multilingual Strings

The instance names and enumeration strings in this object are multilingual, and are translated based on the current language settings as follows:

| Instance | English | German | Spanish | Italian | French |
|---|---|---|---|---|---|
| 3 | IP address | IP-Adresse | Dirección IP | Indirizzo IP | Adresse IP |
| 4 | Subnet mask | Subnetzmaske | Masac. subred | Sottorete | Sous-réseau |
| 5 | Gateway | Gateway | Pasarela | Gateway | Passerelle |
| 6 | DHCP | DHCP | DHCP | DHCP | DHCP |
| | Enable | Einschalten | Activado | Abilitato | Activé |
| | Disable | Ausschalten | Desactivado | Disabilitato | Désactivé |
| 7 | Comm 1 | Komm 1 | Comu 1 | Connessione 1 | Comm 1 |
| 8 | Comm 2 | Komm 2 | Comu 2 | Connessione 2 | Comm 2 |
| 9 | DNS1 | DNS 1 | DNS Primaria | DNS1 | DNS1 |
| 10 | DNS2 | DNS 2 | DNS Secundia. | DNS2 | DNS2 |
| 11 | Host name | Host name | Nombre Host | Nome Host | Nom hôte |
| 12 | Domain name | Domain name | Nobre Domain | Nome Dominio | Dom Domaine |
| 13 | SMTP Server | SMTP Server | Servidor SMTP | Server SMTP | SMTP serveur |
| 14 | SMTP User | SMTP User | Usuario SMTP | Utente SMTP | SMTP utilisa. |
| 15 | SMTP Pswd | SMTP PSWD | Clave SMTP | Password SMTP | SMTP mt passe |
| 20 | Station name | Stationsname | Nom. Estacion | Nome Stazione | Nom Station |
| 21 | F-Address | F-Adresse | Dirección-F | Indirizzo-F | F-Adresse |

## Command Details: Reset

### Category

Extended

### Details

Command Code: 05h

Valid for: Object Instance

### Description

A reset command to this object will result in that all instances are set to their default values.

It is optional to implement support for this command.

- **Command Details**

| Field | Comments |
|---|---|
| CmdExt[0] | 00h (Reserved) |
| CmdExt[1] | 01h (Factory default reset) |

- **Response Details**

(No data)

# 11.6 Network PROFINET IO Object (0Eh)

## General Information

Extended, advanced

When the application maps ADIs to process data during start-up, the Anybus CompactCom 40 will create the module configuration as described in "Real Identification (RI)" on page 23. The modules in the GSDML file must then be described in the same way. The GSDML file provided by HMS provides a few examples based on this way of describing modules.

If the end-user wishes to define modules in another way the application must provide the module configuration to the CompactCom 40. This is achieved by using the following commands:

- API_Add
- Plug_Module
- Plug_Submodule
- Plug_Submodule_Ext

These commands need to be sent after the process data is mapped, and before sending setup complete to the CompactCom 40.

### Example:

Initially, the application maps ADIs as process data by calling all or some of the functions below:

- Map_ADI_Write_Area (10h)
- Map_ADI_Read_Area (11h)
- Map_ADI_Write_Ext_Area (12h)
- Map_ADI_Read_Ext_Area (13h)

The CompactCom 40 now creates modules and submodules based on this information as described in "Real Identification (RI)" on page 23.

To alter this configuration, do the following:

1. Call API_Add to add an API.
2. Call Plug_Module to add a module to the API.
3. Call Plug_Submodule one or more times to add submodules to the module.
4. Repeat steps 2 and 3 to add modules to the API.

After the configuration is complete, call setup complete.

### Removing and exchanging modules and submodules

There are ways of removing modules and plug new modules during runtime.

The end user may have defined five different modules in the GSDML file, and during start-up one of them is always mapped as the default mapping.

When constructing the PLC program the PLC programmer decides to use two different modules as process data.

The application will then be notified by the command Cfg_Mismatch_Ind for every submodule that does not match. This information will also be provided in the command Expected_Ident_Ind. The application can then decide to remove the plugged module by issuing the command Pull_Module. This will remove the whole module and its submodules. Then, based on the information received from either Cfg_Mismatch_Ind or Expected_Ident_Ind, the application can adopt to the PLC configuration by issuing new Plug_Module, Plug_Submodule and Plug_Submodule_Ext commands.

## Object Description

-

## Supported Commands

Object: Get_Attribute
Plug_Module (see "Command Details: Plug_Module" on page 88)
Plug_Submodule (see "Command Details: Plug_Submodule" on page 89)
Plug_Submodule_Ext (see "Command Details: Plug_Submodule_Ext" on page 91)
Pull_Module (see "Command Details: Pull_Module" on page 93)
Pull_Submodule (see "Command Details: Pull_Submodule" on page 94)
API_Add (see "Command Details: API_Add" on page 95)
Appl_State_Ready (see "Command Details: Appl_State_Ready" on page 96)
AR_Abort (see "Command Details: AR_Abort" on page 97)
IM_Options (see "Command Details: IM_Options" on page 98)

Instance: Get_Attribute

## Object Attributes (Instance #0)

| # | Name | Access | Data Type | Value |
|---|---|---|---|---|
| 1 | Name | Get | Array of CHAR | 'Network PROFINET IO' |
| 2 | Revision | Get | UINT8 | 01h |
| 3 | Number of instances | Get | UINT16 | 0001h |
| 4 | Highest instance no. | Get | UINT16 | 0001h |

## Instance Attributes (Instance #1)

### Extended

| # | Name | Access | Type | Value |
|---|---|---|---|---|
| 1 | OnLineTrans | Get | UINT32 | Diagnostic counters; keeps track of the number of on-line transitions |
| 2 | OffLineTrans | Get | UINT32 | Diagnostic counters; keeps track of the number of off-line transitions |
| 3 | (reserved) | | | |
| 4 | Last AbortInd ReasonCode | Get | UINT16 | Reason code for most recent Abort indication<br><br>See also...<br>- “Command Details: AR_Abort” on page 97 |
| 5 | AddedApis | Get | UINT16 | Returns the number of APIs added (including API 0) |
| 6 | ApiList | Get | Array of UINT32 | First element will always be zero and the second element will contain an additional API. Length of the array is determined by parameter “AddedApis”. |
| 7 | EstablishedArs | Get | UINT16 | The number of Application Relationships currently established |
| 8 | ArList | Get | Array of UINT16 | Array of Application Relationship handles. Length of array is determined by parameter “EstablishedArs”. |
| 9 | - | - | - | - |
| 10 | Port 1 MAC Address | Get | Array of UINT8 | 6 Byte PROFINET Port 1 MAC address<br><br>See also...<br>- “Ethernet Host Object (F9h)” on page 145 |
| 11 | Port 2 MAC Address | Get | Array of UINT8 | 6 Byte PROFINET Port 2 MAC address<br><br>See also...<br>- “Ethernet Host Object (F9h)” on page 145 |

# Command Details: Plug_Module

## Category

Advanced

## Details

Command Code.: 10h

Valid for: Object Instance

## Description

This command may be called during start-up to specify the Real Identification. It may also be called during runtime in case there are changes to the Real Identification.

**Note:** It is only permitted to issue this command if 'API_Add' has been issued first.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved, set to zero) | (reserved for future use) |
| CmdExt[1] | | |
| Data[0] | API (low word, low byte) | Application Process Instance (API)<br><br>See also...<br>- "Application Process Instances (API)" on page 22 |
| Data[1] | API (low word, high byte) | |
| Data[2] | API (high word, low byte) | |
| Data[3] | API (high word, high byte) | |
| Data[4] | SlotNr (low byte) | Destination slot for module.<br><br>Range:<br>0... 0x7FFF |
| Data[5] | SlotNr (high byte) | |
| Data[6] | ModIdent (low word, low byte) | Module identified as state in the GSD-file |
| Data[7] | ModIdent (low word, high byte) | |
| Data[8] | ModIdent (high word, low byte) | |
| Data[9] | ModIdent (high word, high byte) | |

See also...

- "Real Identification (RI)" on page 23 ( "Configuration Mismatch" on page 24)
- "Command Details: Plug_Submodule" on page 89
- "Command Details: Pull_Module" on page 93
- "Command Details: Pull_Submodule" on page 94
- "Command Details: API_Add" on page 95

## Command Details: Plug_Submodule

### Category

Advanced

### Details

Command Code.: 11h

Valid for: Object Instance

### Description

This command may be called during start-up to specify the Real Identification. It may also be called during runtime in case there are changes to the Real Identification. In such case, the Anybus will automatically issue a 'Plug' or 'Plug Wrong Submodule'-alarm to the IO Controller.

A submodule plugged with this command can hold IO data to the master, from the master or data in both directions. It is also possible to plug submodules which do not carry any data at all.

The Anybus supports up to 128 submodules in total.

**Note 1:** In case the slot number in the command is set to 0 (zero), the ADI number must also be 0 (zero), since slot 0 cannot hold any actual data.

**Note 2:** It is only permitted to issue this command if 'API_Add' has been issued first.

**Note 3:** The 'Interface'- and 'Port'- submodules have to be plugged in order to pass certification tests.

**Note 4:** The interface and port submodule can only be plugged during the SETUP-state. Any attempt to plug these submodules during runtime will result in error.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved, set to zero) | (reserved for future use) |
| CmdExt[1] | | |
| Data[0] | ADI number (low byte), Read | Reference to the ADI previously mapped with Map_ADI_Read_Area. |
| Data[1] | ADI number (high byte), Read | |
| Data[2] | ADI element, Read | Reference to the element of the ADI mapped with Map_ADI_Read_Area for the specified SlotNr (See Data[10... 11]).<br><br>Range:<br>1... 255ADI element associated with the submodule<br>0 Entire ADI is associated with the submodule |
| Data[3] | ADI number (low byte), Write | Reference to the ADI previously mapped with Map_ADI_Write_Area. |
| Data[4] | ADI number (high byte), Write | |
| Data[5] | ADI element, Read | Reference to the element of the ADI mapped with Map_ADI_Write_Area for the specified SlotNr (See Data[10... 11]).<br><br>Range:<br>1... 255ADI element associated with the submodule<br>0 Entire ADI is associated with the submodule |
| Data[6] | API (low word, low byte) | Application Process Instance (API)<br><br>See also...<br>- "Application Process Instances (API)" on page 22 |
| Data[7] | API (low word, high byte) | |
| Data[8] | API (high word, low byte) | |
| Data[9] | API (high word, high byte) | |
| Data[10] | SlotNr (low byte) | Destination slot for submodule.<br><br>Range:<br>0... 0x7FFF |
| Data[11] | SlotNr (high byte) | |
| Data[12] | SubSlotNr (low byte) | Destination subslot for submodule.<br>Range:<br>For API 0: 1... 0x8002<br>For API >0: 1... 0x7FFF |
| Data[13] | SubSlotNr (high byte) | |
| Data[14] | SubModIdent (low word, low byte) | Submodule identifier as stated in the GSD-file |
| Data[15] | SubModIdent (low word, high byte) | |
| Data[16] | SubModIdent (high word, low byte) | |
| Data[17] | SubModIdent (high word, high byte) | |

See also...

- "Real Identification (RI)" on page 23 ( "Configuration Mismatch" on page 24)
- "Command Details: Plug_Module" on page 88
- "Command Details: Pull_Module" on page 93
- "Command Details: Pull_Submodule" on page 94
- "Command Details: API_Add" on page 95

# Command Details: Plug_Submodule_Ext

## Category

Advanced

## Details

Command Code.: 19h

Valid for: Object Instance

## Description

This is an extended version of the Plug_Submodule command. This command may be called during start-up to specify the Real Identification. It additionally features the possibility to associate a submodule with several consecutive ADI elements. (The Plug_Submodule command only allows association with one ADI element or all ADI elements.)

This command can also be called during operation if there are changes to the Real Identification. A 'Plug'-alarm or a 'Plug Wrong Submodule'-alarm is automatically sent to the master as a result of this action.

A submodule plugged with this command can hold IO data to the master, from the master or data in both directions. It is also possible to plug submodules which do not carry any data at all.

The Anybus CompactCom 40 supports up to 128 submodules in total.

**Note 1:** In case the slot number in the command is set to 0 (zero), the ADI number must also be 0 (zero), since slot 0 cannot hold any actual data.

**Note 2:** It is only permitted to issue this command if 'API_Add' has been issued first.

**Note 3:** The 'Interface'- and 'Port'- submodules have to be plugged in order to pass certification tests.

**Note 4:** The interface and port submodule can only be plugged during the SETUP-state. Any attempt to plug these submodules during runtime will result in error.

**Note 5:** It is not recommended to mix Plug_Submodule and Plug_Submodule_Ext commands.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved, set to zero) | (reserved for future use) |
| CmdExt[1] | | |
| Data[0] | ADI number (low byte), Read | Reference to the ADI previously mapped with Map_ADI_Read_Area. |
| Data[1] | ADI number (high byte), Read | |
| Data[2] | First ADI element, Read | Reference to the first ADI element associated with the submodule.<br><br>Range:<br>0 to 255 - ADI element (0 is the first element) |
| Data[3] | Number of consecutive ADI elements, Read | Number of consecutive elements associated with the submodule.<br><br>Range:<br>1 to 255 (Number of elements) |
| Data[4] | ADI number (low byte), Write | Reference to the ADI previously mapped with Map_ADI_Write_Area. |
| Data[5] | ADI number (high byte), Write | |
| Data[6] | ADI element, Write | Reference to the first ADI element associated with the submodule.<br><br>Range:<br>0 to 255 - ADI element (0 is the first element) |
| Data[7] | Number of consecutive ADI elements, Write | Number of consecutive elements associated with the submodule.<br><br>Range:<br>1 to 255 (Number of elements) |
| Data[8] | API (low word, low byte) | Application Process Instance (API)<br><br>See also...<br>- "Application Process Instances (API)" on page 22 |
| Data[9] | API (low word, high byte) | |
| Data[10] | API (high word, low byte) | |
| Data[11] | API (high word, high byte) | |
| Data[12] | SlotNr (low byte) | Destination slot for submodule.<br><br>Range:<br>0... 0x7FFF |
| Data[13] | SlotNr (high byte) | |
| Data[14] | SubSlotNr (low byte) | Destination subslot for submodule.<br>Range:<br>For API 0: 1... 0x8002<br>For API >0: 1... 0x7FFF |
| Data[15] | SubSlotNr (high byte) | |
| Data[16] | SubModIdent (low word, low byte) | Submodule identifier as stated in the GSD file |
| Data[17] | SubModIdent (low word, high byte) | |
| Data[18] | SubModIdent (high word, low byte) | |
| Data[19] | SubModIdent (high word, high byte) | |

See also...

- "Real Identification (RI)" on page 23 ( "Configuration Mismatch" on page 24)
- "Command Details: Plug_Module" on page 88
- "Command Details: Pull_Module" on page 93
- "Command Details: Pull_Submodule" on page 94
- "Command Details: API_Add" on page 95

# Command Details: Pull_Module

## Category

Advanced

## Details

Command Code.: 12h

Valid for: Object Instance

## Description

This command removes a module from the configuration. Can be issued at any time. During runtime, it can be called in case there are changes to the Real Identification. The CompactCom 40 then automatically issues a 'Pull' or 'Pull Module' alarm to the master.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved, set to zero) | (reserved for future use) |
| CmdExt[1] | | |
| Data[0] | API (low word, low byte) | Application Process Instance (API)<br><br>See also...<br>- "Application Process Instances (API)" on page 22 |
| Data[1] | API (low word, high byte) | |
| Data[2] | API (high word, low byte) | |
| Data[3] | API (high word, high byte) | |
| Data[4] | SlotNr (low byte) | Slot number of module.<br><br>Range:<br>0... 0x7FFF |
| Data[5] | SlotNr (high byte) | |

See also...

- "Command Details: Plug_Module" on page 88
- "Command Details: Pull_Submodule" on page 94

# Command Details: Pull_Submodule

## Category

Advanced

## Details

Command Code.: 13h

Valid for: Object Instance

## Description

This command removes a submodule from the configuration. Can be issued at any time. During run-time, it can be called in case there are changes to the Real Identification. The CompactCom 40 then automatically issues a 'Pull' alarm to the master.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved, set to zero) | (reserved for future use) |
| CmdExt[1] | | |
| Data[0] | API (low word, low byte) | Application Process Instance (API)<br><br>See also...<br>- "Application Process Instances (API)" on page 22 |
| Data[1] | API (low word, high byte) | |
| Data[2] | API (high word, low byte) | |
| Data[3] | API (high word, high byte) | |
| Data[4] | SlotNr (low byte) | Slot number of submodule.<br><br>Range:<br>0... 0x7FFF |
| Data[5] | SlotNr (high byte) | |
| Data[6] | SubSlotNr (low byte) | Subslot number of submodule.<br>Range:<br>For API 0: 1... 0x8002<br>For API >0: 1... 0x7FFF |
| Data[7] | SubSlotNr (high byte) | |

See also...

- "Command Details: Plug_Submodule" on page 89
- "Command Details: Pull_Module" on page 93

# Command Details: API_Add

## Category

Advanced

## Details

Command Code.: 14h

Valid for: Object Instance

## Description

By default, the module only supports API 0 (zero). If additional APIs are to be supported, or if the host application shall handle plugging/unplugging of modules and submodules, this command must be used to specify the API implementation. Note that if using this command, it is mandatory to declare API 0 (zero) prior to defining other APIs or plugging/unplugging modules/submodules. API numbers are assigned by (PROFIBUS & PROFINET International (PI)).

**Note 1:** This command may only be issued prior to setting the 'Setup Complete'-attribute in the Anybus Object.

**Note 2:** This command clears the default Real Identification created by the Anybus module while mapping ADIs to Process Data. Therefore, issuing this command effectively makes it mandatory to specify the actual Real Identification by means of the 'Plug_Module' and 'Plug_Submodule'-commands.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved, set to zero) | (reserved for future use) |
| CmdExt[1] | | |
| Data[0] | API (low word, low byte) | Application Process Instance (API)<br><br>See also...<br>- "Application Process Instances (API)" on page 22 |
| Data[1] | API (low word, high byte) | |
| Data[2] | API (high word, low byte) | |
| Data[3] | API (high word, high byte) | |
| Data[4] | (reserved, set to zero) | (reserved) |
| Data[5] | | |
| Data[6] | | |
| Data[7] | | |

See also...

- "Application Process Instances (API)" on page 22

# Command Details: Appl_State_Ready

## Category

Advanced

## Details

Command Code.: 15h

Valid for: Object Instance

## Description

This command is only applicable if the host application implements support for 'End_Of_Prm_Ind', and signals to the module (and in turn the I/O Controller) that the host application is ready for data exchange.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | AR handle (low byte) | Handle for the Application Relationship (AR) (see "Application Process Instances (API)" on page 22) |
| CmdExt[1] | AR handle (high byte) | |

See also...

- "Application Relationships (AR)" on page 22
- "Command Details: End_Of_Prm_Ind" on page 142

# Command Details: AR_Abort

## Category

Advanced

## Details

Command Code.: 16h

Valid for: Object Instance

## Description

This command indicates to the module that the current application relationship shall be aborted.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | AR handle (low byte) | Handle for the Application Relationship (AR) (see "Application Process Instances (API)" on page 22) |
| CmdExt[1] | AR handle (high byte) | |

See also...

- "Application Relationships (AR)" on page 22
- "Command Details: AR_Check_Ind" on page 138
- "Command Details: Expected_Ident_Ind" on page 140
- ""Command Details: Appl_State_Ready" on page 96" on page 142

# Command Details: IM_Options

## Category

Advanced

## Details

Command Code.: 18h

Valid for: Object Instance

## Description

During startup, this command can be called to specify if I&M data for Slot0 (DAP) and/or Slot > 0 should be forwarded transparently by the Anybus CompactCom module. Additionally, it provides a way for the application to specify the I&M0 Filter Data when I&M data is transparent.

I&M0 Filter Data is composed of three blocks: I&M0 Carrier Data, Module Representative Data and Device Representative Data (see table below). A submodule can belong to several blocks.

| I&M0 Filter Data Contents | |
|---|---|
| **Content** | **Description** |
| I&M0 Carrier Data | List of all submodules being a carrier of discrete I&M data<br>Block is mandatory to support according to the PROFINET specification |
| Module Representative Data | List of all submodules acting as module representative<br>Block is optional to support according to the PROFINET specification |
| Device Representative Data | List of at least one submodule where I&M1, I&M2, I&M3 and I&M4 can be written<br>Block is mandatory to support according to the PROFINET specification |

When transparent I&M data for slot > 0 is enabled, the application must store I&M data for modules located in slots > 0 to nonvolatile memory. In this case, all modules that carry discrete I&M data shall be included in the 'I&M0 Carrier Data'. (The Anybus CompactCom 40 module will include the DAP submodule (located in slot 0, subslot 1) in the 'I&M0 Carrier Data' and 'Device Representative Data'.)

When transparent I&M data for slot 0 is enabled, the application must store I&M data for slot 0 to non-volatile memory. The DAP submodule is by default included in the 'I&M0 Carrier Data' and 'Device Representative Data' but may be removed from any of the blocks using the IM_Options command.

See also...

- "Flowchart - I&M Record Data Handling (API = 0)" on page 153.
- "Command Details: Get_Record" on page 134
- "Command Details: Set_Record" on page 136

- **Command Details**

  If the I&M0 Filter Data is of no interest, the Data Field is left out (command length = 0).

  The command may contain one or several I&M0 Filter Data entries. The maximum amount of entries depends on the application. For a 256 bytes message channel the maximum amount of entries is 51 (256 / 5 = 51). For a 1524 bytes message channel, the maximum amount of entries is 304 (1524 / 5 = 304).

  **Important:** For submodules to be listed in the I&M0 Filter data, the command must be sent when the Real Identification has been determined. This means that for the "ADI Based RI" method, the command must be sent when the module has shifted to WAIT_PROCESS state. For the "Application specific RI" method the command can be sent in SETUP state but after the plugging of modules/submodules is finished (Plug_Module/Plug_Submodule).

  The table below contains two entries as an example.

| Field | Contents | Comments | Example |
|---|---|---|---|
| CmdExt[0] | IM_Transparent | Bitmask for enabling transparent I&M data See table below | 02h |
| CmdExt[1] | Reserved | | |
| Data[0] | SlotNr (low byte) | Location of the submodule that shall be part of the I&M0 Filter Data | Entry #1 - module with discrete I/M data in slot 1, subslot 1 |
| Data[1] | SlotNr (high byte) | | |
| Data[2] | SubSlotNr (low byte) | | |
| Data[3] | SubSlotNr (high byte) | | |
| Data[4] | IM0_Filter_Data | Bitmask specifying which I&M0 Filter Data block(s) the submodule shall belong to See table below | 01h |
| Data[5] | SlotNr (low byte) | | Entry #2 - module with discrete I&M data in slot 2, subslot 1 |
| Data[6] | SlotNr (high byte) | | |
| Data[7] | SubSlotNr (low byte) | | |
| Data[8] | SubSlotNr (high byte) | | |
| Data[9] | IM0_Filter_Data | | 01h |

| IM_Transparent | | |
|---|---|---|
| **Bit** | **Value** | **Description** |
| 0 | 0 | |
| | 1 | Transparent I&M data for SlotNr = 0 (DAP) |
| 1 | 0 | |
| | 1 | Transparent I&M data for SlotNr not equal to 0 |
| 2 - 7 | - | Reserved |

| IM0_Filter_Data | | |
|---|---|---|
| **Bit** | **Value** | **Description** |
| 0 | 0 | |
| | 1 | Submodule will be included in the I&M0 Carrier Data block |
| 1 | 0 | |
| | 1 | Submodule will be included in the Device Representative Data block |
| 2 | 0 | |
| | 1 | Submodule will be included in the Module Representative Data block |
| 3 - 7 | - | Reserved |

## Object Specific Error Codes

| Code | Meaning |
|---|---|
| 01h | The ADI has not been mapped with command Map_ADI_Write_Area |
| 02h | The ADI has not been mapped with command Map_ADI_Read_Area |
| 03h | Element does not exist for the ADI |
| 04h | This ADI/element is already mapped |
| 05h | API 0 must be added first |
| 06h | API does not exist |
| 07h | Trying to add an API already present |
| 08h | There is no room for any more APIs |
| 09h | Module in slot 0 cannot have any IO data |
| 0Ah | Prior to plugging the requested module/submodule, slot 0 must be populated with a module and a submodule (in subslot 1) |
| 0Bh | Slot occupied |
| 0Ch | subslot occupied |
| 0Dh | No module inserted in the specified slot |
| 0Eh | No submodule inserted in the specified subslot |
| 0Fh | Slot number specified is out-of-range |
| 10h | subslot number specified is out-of-range |
| 11h | The AR handle provided is not valid |
| 12h | There is no application ready pending |
| 13h | Unknown error (PROFINET IO stack denied the request) |
| 14h | Max number of submodules have already been plugged |
| 15h | Safety module has not been plugged |
| 16h | ADI data type constraint |

# 11.7 Socket Interface Object (07h)

## Category

Advanced

## Object Description

This object provides direct access to the TCP/IP stack socket interface, enabling custom protocols to be implemented over TCP/UDP.

Note that some of the commands used when accessing this object may require segmentation. For more information, see "Message Segmentation" on page 150.

---

**IMPORTANT:** *The use of functionality provided by this object should only be attempted by users who are already familiar with socket interface programming and who fully understands the concepts involved in TCP/IP programming.*

## Supported Commands

Object: Get_Attribute
Create (See "Command Details: Create" on page 103)
Delete (See "Command Details: Delete" on page 104)

Instance: Get_Attribute
Set_Attribute
Bind (See "Command Details: Bind" on page 105)
Shutdown (See "Command Details: Shutdown" on page 106)
Listen (See "Command Details: Listen" on page 107)
Accept (See "Command Details: Accept" on page 108)
Connect (See "Command Details: Connect" on page 109)
Receive (See "Command Details: Receive" on page 110)
Receive_From (See "Command Details: Receive_From" on page 111)
Send (See "Command Details: Send" on page 112)
Send_To (See "Command Details: Send_To" on page 113)
IP_Add_membership (See "Command Details: IP_Add_Membership" on page 114)
IP_Drop_membership (See "Command Details: IP_Drop_Membership" on page 115)
DNS_Lookup (See "Command Details: DNS_Loopup" on page 116)

## Object Attributes (Instance #0)

| # | Name | Access | Data Type | Value |
|---|---|---|---|---|
| 1 | Name | Get | Array of CHAR | 'Socket interface' |
| 2 | Revision | Get | UINT8 | 01h |
| 3 | Number of instances | Get | UINT16 | - |
| 4 | Highest instance no. | Get | UINT16 | - |
| 11 | Max. no. of instances | Get | UINT16 | 0008h |

# Instance Attributes (Sockets #1...8)

## Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | Socket type | Get | UINT8 | Value:Socket Type:<br>00h SOCK_STREAM, NONBLOCKING (TCP)<br>01h SOCK_STREAM, BLOCKING (TCP)<br>02h SOCK_DGRAM, NONBLOCKING (UDP)<br>03h SOCK_DGRAM, BLOCKING (UDP) |
| 2 | Port | Get | UINT16 | Local port that the socket is bound to |
| 3 | Host IP | Get | UINT32 | Host IP address, or 0 (zero) if not connected |
| 4 | Host port | Get | UINT16 | Host port number, or 0 (zero) if not connected |
| 5 | TCP State | Get | UINT8 | State (TCP sockets only):<br>Value:State:Description:<br>00h CLOSED Closed<br>01h LISTEN Listening for connection<br>02h SYN_SENT Active, have sent SYN<br>03h SYN_RECEIVED Have sent and received SYN<br>04h ESTABLISHED Established.<br>05h CLOSE_WAIT Received FIN, waiting for close<br>06h FIN_WAIT_1 Have closed, sent FIN<br>07h CLOSING Closed exchanged FIN; await FIN ACK<br>08h LAST_ACK Have FIN and close; await FIN ACK<br>09h FIN_WAIT_2 Have closed, FIN is acknowledged<br>0Ah TIME_WAIT Quiet wait after close |
| 6 | TCP RX bytes | Get | UINT16 | Number of bytes in RX buffers (TCP sockets only) |
| 7 | TCP TX bytes | Get | UINT16 | Number of bytes in TX buffers (TCP sockets only) |
| 8 | Reuse address | Get/Set | BOOL | Socket can reuse local address<br>Value:Meaning:<br>1 Enabled<br>0 Disabled (default) |
| 9 | Keep alive | Get/Set | BOOL | Protocol probes idle connection (TCP sockets only).[a]<br>Value:Meaning:<br>1 Enabled<br>0 Disabled (default) |
| 10 | IP Multicast TTL | Get/Set | UINT8 | IP Multicast TTL value (UDP sockets only).<br>Default = 1. |
| 11 | IP Multicast Loop | Get/Set | BOOL | IP multicast loop back (UDP sockets only)[b]<br>Value:Meaning:<br>1 Enable (default)<br>0 Disable |
| 12 | Ack delay time | Get/Set | UINT16 | Time for delayed ACKs in ms (TCP sockets only)<br>Default = 200ms[c] |
| 13 | TCP No Delay | Get/Set | BOOL | Don't delay send to coalesce packets (TCP).<br>Value:Meaning:<br>1 Delay (default)<br>0 Don't delay (turn off Nagle's algorithm on socket) |
| 14 | TCP Connect Timeout | Get/Set | UINT16 | TCP Connect timeout in seconds (default = 75s) |

a. If the Keep alive attribute is set, the connection will be probed for the first time after it has been idle for 120 minutes. If a probe attempt fails, the connection will continue to be probed at intervals of 75 s. The connection is terminated after 8 failed probe attempts.
b. Must belong to group in order to get the loop backed message
c. Resolution is 50ms, i.e. 50...99 = 50ms, 100...149 = 100ms, 199 = 150ms etc.

# Command Details: Create

## Category

Advanced

## Details

Command Code.: 03h

Valid for: Object Instance

## Description

This command creates a socket.

**Note:** This command is only allowed in WAIT_PROCESS, IDLE and PROCESS_ACTIVE states.

- **Command Details**

| Field | Contents |
|---|---|
| CmdExt[0] | (reserved, set to zero) |
| CmdExt[1] | Value:Socket Type:<br>00h SOCK_STREAM, NON-BLOCKING (TCP)<br>01h SOCK_STREAM, BLOCKING (TCP)<br>02h SOCK_DGRAM, NON-BLOCKING (UDP)<br>03h SOCK_DGRAM, BLOCKING (UDP) |

- **Response Details**

| Field | Contents | Comments |
|---|---|---|
| Data[0] | Instance number (low) | Instance number of the created socket. |
| Data[1] | Instance number (high) | |

# Command Details: Delete

## Category

Advanced

## Details

Command Code.: 04h

Valid for: Object Instance

## Description

This command deletes a previously created socket and closes the connection (if connected).

- If the socket is of TCP-type and a connection is established, the connection is terminated with the RST-flag.
- To gracefully terminate a TCP-connection, it is recommended to use the 'Shutdown'-command (see "Command Details: Shutdown" on page 106) before deleting the socket, causing the connection to be closed with the FIN-flag instead.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | Instance number to delete (low) | Instance number of socket that shall be deleted. |
| CmdExt[1] | Instance number to delete (high) | |

- **Response Details**

  (no data)

# Command Details: Bind

## Category

Advanced

## Details

Command Code.: 10h

Valid for: Instance

## Description

This command binds a socket to a local port.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | Requested port number (low) | Set to 0 (zero) to request binding to any free port. |
| CmdExt[1] | Requested port number (high) | |

- **Response Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | Bound port number (low) | Actual port that the socket was bound to. |
| CmdExt[1] | Bound port number (high) | |

# Command Details: Shutdown

## Category

Advanced

## Details

Command Code.: 11h

Valid for: Instance

## Description

This command closes a TCP-connection using the FIN-flag. Note that the response does not indicate if the connection actually shut down, which means that this command cannot be used to poll non-blocking sockets, nor will it block for blocking sockets.

- **Command Details**

| Field | Contents |
|---|---|
| CmdExt[0] | (reserved, set to zero) |
| CmdExt[1] | Value:Mode:<br>00h Shutdown receive channel<br>01h Shutdown send channel<br>02h Shutdown both receive- and send channel |

- **Response Details**

  (no data)

The recommended sequence to gracefully shut down a TCP connection is described below.

Application initiates shutdown:

**1.** Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send channel, note that the receive channel will still be operational.

**2.** Receive data on socket until error message Object specific error (EDESTADDRREQ (14)) is received, indicating that the host closed the receive channel. If host does not close the receive channel use a timeout and progress to step 3.

**3.** Delete the socket instance. If step 2 timed out, RST-flag will be sent to terminate the socket.

Host initiates shutdown:

**1.** Receive data on socket, if zero bytes received it indicates that the host closed the receive channel of the socket.

**2.** Try to send any unsent data to the host.

**3.** Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the receive channel.

**4.** Delete the socket instance.

# Command Details: Listen

## Category

Advanced

## Details

Command Code.: 12h

Valid for: Instance

## Description

This command puts a TCP socket in listening state. Backlog queue length is the number of unaccepted connections allowed on the socket. When backlog queue is full, further connections will be refused with RST-flag.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Value:Backlog queue length:<br>00h 1<br>01h 2<br>02h 4 | - |

- **Response Details**

  (no data)

# Command Details: Accept

## Category

Advanced

## Details

Command Code.: 13h

Valid for: Instance

## Description

This command accepts incoming connections on a listening TCP socket. A new socket instance is created for each accepted connection. The new socket is connected with the host and the response returns its instance number.

NONBLOCKING mode:

This command must be issued repeatedly (polled) for incoming connections. If no incoming connection request exists, the module will respond with error code 0006h (EWOULDBLOCK).

BLOCKING mode:

This command will block until a connection request has been detected.

**Note:** This command will only be accepted if there is a free instance to use for accepted connections. For blocking connections, this command will reserve an instance.

- **Command Details**

  (no data)

- **Response Details**

| Field | Contents |
|---|---|
| Data[0] | Instance number for the connected socket (low) |
| Data[1] | Instance number for the connected socket (high) |
| Data[2] | Host IP address byte 3 (low) |
| Data[3] | Host IP address byte 2 |
| Data[4] | Host IP address byte 1 |
| Data[5] | Host IP address byte 0 (high) |
| Data[6] | Host port number (low) |
| Data[7] | Host port number (high) |

# Command Details: Connect

## Category

Advanced

## Details

Command Code.: 14h

Valid for: Instance

## Description

For SOCK-DGRAM-sockets, this command specifies the peer with which the socket is to be associated (to which datagrams are sent and the only address from which datagrams are received).

For SOCK_STREAM-sockets, this command attempts to establish a connection to a host.

SOCK_STREAM-sockets may connect successfully only once, while SOCK_DGRAM-sockets may use this service multiple times to change their association. SOCK-DGRAM-sockets may dissolve their association by connecting to IP address 0.0.0.0, port 0 (zero).

NON-BLOCKING mode:

This command must be issued repeatedly (polled) until a connection is connected, rejected or timed out. The first connect-attempt will be accepted, thereafter the command will return error code 22 (EINPROGRESS) on poll requests while attempting to connect.

BLOCKING mode:

This command will block until a connection has been established or the connection request is cancelled due to a timeout or a connection error.

- **Command Details**

| Field | Contents | Contents |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | | |
| Data[0] | Host IP address byte 3 (low) | - |
| Data[1] | Host IP address byte 2 | |
| Data[2] | Host IP address byte 1 | |
| Data[3] | Host IP address byte 0 (high) | |
| Data[4] | Host port number (low) | |
| Data[5] | Host port number (high) | |

- **Response Details**

  (no data)

# Command Details: Receive

## Category

Advanced

## Details

Command Code.: 15h

Valid for: Instance

## Description

This command receives data from a connected socket. Message segmentation may be used to receive up to 1472 bytes (see "Message Segmentation" on page 150).

For SOCK-DGRAM-sockets, the module will return the requested amount of data from the next received datagram. If the datagram is smaller than requested, the entire datagram will be returned in the response message. If the datagram is larger than requested, the excess bytes will be discarded.

For SOCK_STREAM-sockets, the module will return the requested number of bytes from the received data stream. If the actual data size is less than requested, all available data will be returned.

NON-BLOCKING mode:

If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.

BLOCKING mode:

The module will not issue a response until the operation has finished.

If the module responds successfully with 0 (zero) bytes of data, it means that the host has closed the connection. The send channel may however still be valid and must be closed using 'Shutdown' and/or 'Delete'.

- **Command Details**

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Segmentation Control bits | see "Command Segmentation" on page 150 |
| Data[0] | Receive data size (low) | Only used in the first segment |
| Data[1] | Receive data size (high) | |

- **Response Details**

**Note:** The data in the response may be segmented (see "Message Segmentation" on page 150).

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Segmentation Control bits | see "Response Segmentation" on page 151 |
| Data[0...n] | Received data | - |

# Command Details: Receive_From

## Category

Advanced

## Details

Command Code.: 16h

Valid for: Instance

## Description

This command receives data from an unconnected SOCK_DGRAM-socket. Message segmentation may be used to receive up to 1472 bytes (see "Message Segmentation" on page 150).

The module will return the requested amount of data from the next received datagram. If the datagram is smaller than requested, the entire datagram will be returned in the response message. If the datagram is larger than requested, the excess bytes will be discarded.

The response message contains the IP address and port number of the sender.

NON-BLOCKING mode:

If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.

BLOCKING mode:

The module will not issue a response until the operation has finished.

- **Command Details**

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Segmentation Control bits | see "Command Segmentation" on page 150 |
| Data[0] | Receive data size (low) | Only used in the first segment |
| Data[1] | Receive data size (high) | |

- **Response Details**

**Note:** The data in the response may be segmented (see "Message Segmentation" on page 150).

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Segmentation Control bits | see "Response Segmentation" on page 151 |
| Data[0] | Host IP address byte 3 (low) | The host address/port information is only included in the first segment. All data thereafter will start at Data[0] |
| Data[1] | Host IP address byte 2 | |
| Data[2] | Host IP address byte 1 | |
| Data[3] | Host IP address byte 0 (high) | |
| Data[4] | Host port number (low) | |
| Data[5] | Host port number (high) | |
| Data[6...n] | Received data | |

# Command Details: Send

## Category

Advanced

## Details

Command Code.: 17h

Valid for: Instance

## Description

This command sends data on a connected socket. Message segmentation may be used to send up to 1472 bytes (see "Message Segmentation" on page 150).

NONBLOCKING mode:

If there isn't enough buffer space available in the send buffers, the module will respond with error code 0006h (EWOULDBLOCK)

BLOCKING mode:

If there isn't enough buffer space available in the send buffers, the module will block until there is.

- **Command Details**

  **Note:** To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (see "Message Segmentation" on page 150).

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Segmentation Control | see "Command Segmentation" on page 150 |
| Data[0...n] | Data to send | - |

- **Response Details**

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (ignore) |
| CmdExt[1] | | |
| Data[0] | Number of sent bytes (low) | Only valid in the last segment |
| Data[1] | Number of sent bytes (high) | |

# Command Details: Send_To

## Category

Advanced

## Details

Command Code.: 18h

Valid for: Instance

## Description

This command sends data to a specified host on an unconnected SOCK-DGRAM-socket. Message segmentation may be used to send up to 1472 bytes (see "Message Segmentation" on page 150).

- **Command Details**

  **Note:** To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (see "Message Segmentation" on page 153).

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Segmentation Control | see "Command Segmentation" on page 150 |
| Data[0] | Host IP address byte 3 (low) | The host address/port information shall only be included in the first segment. All data thereafter must start at Data[0] |
| Data[1] | Host IP address byte 2 | |
| Data[2] | Host IP address byte 1 | |
| Data[3] | Host IP address byte 0 (high) | |
| Data[4] | Host port number (low) | |
| Data[5] | Host port number (high) | |
| Data[6...n] | Data to send | |

- **Response Details**

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (ignore) |
| CmdExt[1] | | |
| Data[0] | Number of sent bytes (low) | Only valid in the last segment |
| Data[1] | Number of sent bytes (high) | |

# Command Details: IP_Add_Membership

## Category

Advanced

## Details

Command Code.: 19h

Valid for: Instance

## Description

This command assigns the socket an IP multicast group membership. The module always joins the 'All hosts group' automatically, however this command may be used to specify up to 20 additional memberships.

- **Command Details**

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | | |
| Data[0] | Group IP address byte 3 (low) | - |
| Data[1] | Group IP address byte 2 | |
| Data[2] | Group IP address byte 1 | |
| Data[3] | Group IP address byte 0 (high) | |

- **Response Details**

  (no data)

# Command Details: IP_Drop_Membership

## Category

Advanced

## Details

Command Code.: 1Ah

Valid for: Instance

## Description

This command removes the socket from an IP multicast group membership.

- **Command Details**

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | | |
| Data[0] | Group IP address byte 3 (low) | - |
| Data[1] | Group IP address byte 2 | |
| Data[2] | Group IP address byte 1 | |
| Data[3] | Group IP address byte 0 (high) | |

- **Response Details**

  (no data)

# Command Details: DNS_Loopup

## Category

Advanced

## Details

Command Code.: 1Bh

Valid for: Object Instance

## Description

This command resolves the given host name and returns the IP address.

- **Command Details**

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | | |
| Data[0... N] | Host name | Host name to resolve |

- **Response Details (Success)**

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | | |
| Data[0] | IP address byte 3 (low) | IP address of the specified host |
| Data[1] | IP address byte 2 | |
| Data[2] | IP address byte 1 | |
| Data[3] | IP address byte 0 (high) | |

## Socket Interface Error Codes (Object Specific)

The following object-specific error codes may be returned by the module when using the socket interface object.

| Error Code | Name | Meaning |
|---|---|---|
| 1 | ENOBUFS | No internal buffers available |
| 2 | ETIMEDOUT | A timeout event occurred |
| 3 | EISCONN | Socket already connected |
| 4 | EOPNOTSUPP | Service not supported |
| 5 | ECONNABORTED | Connection was aborted |
| 6 | EWOULDBLOCK | Socket cannot block because unblocking socket type |
| 7 | ECONNREFUSED | Connection refused |
| 8 | ECONNRESET | Connection reset |
| 9 | ENOTCONN | Socket is not connected |
| 10 | EALREADY | Socket is already in requested mode |
| 11 | EINVAL | Invalid service data |
| 12 | EMSGSIZE | Invalid message size |
| 13 | EPIPE | Error in pipe |
| 14 | EDESTADDRREQ | Destination address required |
| 15 | ESHUTDOWN | Socket has already been shutdown |
| 16 | (reserved) | - |
| 17 | EHAVEOOB | Out of band data available |
| 18 | ENOMEM | No internal memory available |
| 19 | EADDRNOTAVAIL | Address is not available |
| 20 | EADDRINUSE | Address already in use |
| 21 | (reserved) | - |
| 22 | EINPROGRESS | Service already in progress |
| 28 | ETOOMANYREFS | Too many references |
| 101 | Command aborted | If a command is blocking on a socket, and that socket is closed using the Delete command, this error code will be returned to the blocking command. |

# 11.8 SMTP Client Object (09h)

## Category

Advanced

## Object Description

This object groups functions related to the SMTP client.

See also...

- "File System" on page 19
- "E-mail Client" on page 39
- "Instance Attributes (Instance #13, SMTP Server)" on page 80
- "Instance Attributes (Instance #14, SMTP User)" on page 81
- "Instance Attributes (Instance #15, SMTP Password)" on page 81

## Supported Commands

| | |
|---|---|
| Object: | Get_Attribute<br>Create<br>Delete<br>Send e-mail from file( "Command Details: Send E-mail From File" on page 121) |
| Instance: | Get_Attribute<br>Set_Attribute<br>Send e-mail( "Command Details: Send E-mail" on page 122) |

## Object Attributes (Instance #0)

| # | Name | Access | Data Type | Value |
|---|---|---|---|---|
| 1 | Name | Get | Array of CHAR | 'SMTP Client' |
| 2 | Revision | Get | UINT8 | 01h |
| 3 | Number of instances | Get | UINT16 | - |
| 4 | Highest instance no. | Get | UINT16 | - |
| 11 | Max. no. of instances | Get | UINT16 | 0006h |
| 12 | Success count | Get | UINT16 | Reflects the no. of successfully sent messages |
| 13 | Error count | Get | UINT16 | Reflects the no. of messages that could not be delivered |

## Instance Attributes

### Advanced

Instances are created dynamically by the application.

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | From | Get/Set | Array of CHAR | e.g. “someone@somewhere.com” |
| 2 | To | Get/Set | Array of CHAR | e.g. “someone.else@anywhere.net” |
| 3 | Subject | Get/Set | Array of CHAR | e.g. “Important notice” |
| 4 | Message | Get/Set | Array of CHAR | e.g. “Duck and cover” |

## Command Details: Create

### Category

Advanced

### Details

Command Code.: 03h

Valid for: Object

### Description

This command creates an e-mail instance.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved, set to zero) | - |
| CmdExt[1] | | |

- **Response Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved, ignore) | - |
| CmdExt[1] | | |
| MsgData[0] | Instance number | low byte |
| MsgData[1] | | high byte |

# Command Details: Delete

## Category

Advanced

## Details

Command Code.: 04h

Valid for: Object

## Description

This command deletes an e-mail instance.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | E-mail instance number | low byte |
| CmdExt[1] | | high byte |

- **Response Details**

  (no data)

# Command Details: Send E-mail From File

## Category

Advanced

## Details

Command Code.: 11h

Valid for: Object

## Description

This command sends an e-mail based on a file in the file system.

File format:

The file must be a plain ASCII-file in the following format:

```
[To]
recipient

[From]
sender

[Subject]
email subject

[Headers]
extra headers, optional

[Message]
actual email message
```

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved, set to zero) | - |
| CmdExt[1] | | |
| MsgData[0... n] | Path + filename of message file | - |

- **Response Details**

(no data)

## Command Details: Send E-mail

### Category

Advanced

### Details

Command Code.: 10h

Valid for: Instance

### Description

This command sends the specified e-mail instance.

- **Command Details**
  (no data)
- **Response Details**
  (no data)

## Object Specific Error Codes

| Error Codes | Meaning |
|---|---|
| 1 | SMTP server not found |
| 2 | SMTP server not ready |
| 3 | Authentication error |
| 4 | SMTP socket error |
| 5 | SSI scan error |
| 6 | Unable to interpret e-mail file |
| 255 | Unspecified SMTP error |
| (other) | (reserved) |

# 11.9 File System Interface Object (0Ah)

## Category

Advanced

## Object Description

This object provides an interface to the built-in file system. Each instance represents a handle to a file stream and contains services for file system operations.

This object is thoroughly described in Anybus CompactCom 40 Software Design Guide.

# 11.10 Network Ethernet Object (0Ch)

## Category

Advanced

## Object Description

This object provides Ethernet-specific information to the application.

## Supported Commands

Object: Get_Attribute

Instance: Get_Attribute

## Object Attributes (Instance #0)

| # | Name | Access | Data Type | Value |
|---|---|---|---|---|
| 1 | Name | Get | Array of CHAR | 'Network Ethernet' |
| 2 | Revision | Get | UINT8 | 01h |
| 3 | Number of instances | Get | UINT16 | - |
| 4 | Highest instance no. | Get | UINT16 | - |

## Instance Attributes (Instance #1)

### Advanced

| # | Name | Access | Type | Description |
|---|---|---|---|---|
| 1 | MAC Address | Get | Array of UINT8 | Current MAC address.<br>See also "Ethernet Host Object (F9h)" on page 145) |
| 2 | Port 1 MAC Address | Get | Array of UINT8 | MAC address for port 1 (mandatory for the LLDP protocol)<br>See also "Ethernet Host Object (F9h)" on page 145) |
| 3 | Port 2 MAC Address | Get | Array of UINT8 | MAC address for port 2 (mandatory for the LLDP protocol)<br>See also "Ethernet Host Object (F9h)" on page 145) |

# 12. Host Application Objects

## 12.1 General Information

This chapter specifies the host application object implementation in the module. The objects listed here may be implemented within the host application firmware to expand the PROFINET implementation.

Standard Objects:

- "Application Object (FFh)" (see Anybus CompactCom 40 Software Design Guide)
- "Application Data Object (FEh)" (see Anybus CompactCom 40 Software Design Guide)
- "Sync Object (EEh)" on page 126
- "Application File System Object (EAh)" (see Anybus CompactCom 40 Software Design Guide)

Network Specific Objects:

- "PROFINET IO Object (F6h)" on page 130
- "Ethernet Host Object (F9h)" on page 145

# 12.2 Sync Object (EEh)

## Category

Extended

## Object Description

This object contains the host object sync settings.

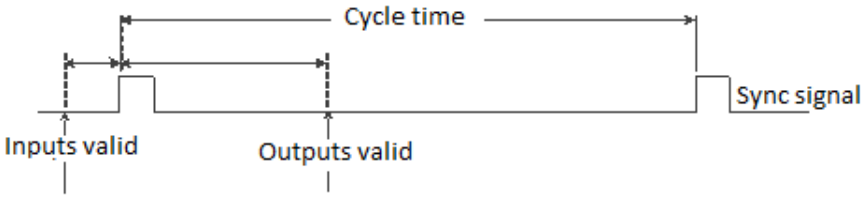
## How to use Sync Functionality

Please note that it is up to the application to make sure that the product is synchronous. The module will only ask the application for:

- **Output processing time**. The time in ns it takes from that the application is notified of new process data until valid output.
- **Input processing time**. The time in ns from input capture time until the process data has been copied to the module.

The module will notify the application of:

- **Output valid**. This is the time in ns relative to the sync event, for when the outputs shall be used.
- **Input capture**. This is the time in ns relative to the sync event, for when the input process data shall be captured.




All modules have the sync signal in the connector and as an interrupt to the host application.

### Calculating Output Processing Time

This needs to be measured by the application designer. The time consists of two delays:

- The delay added by the module, from when message is available on the network until it is available for the host application.
- The delay added by the application. This is the time it takes for the application from it is notified that new process data has arrived, to when the process data is copied and output is valid.

To calculate the delay added by the module the following formula shall be used:

$t_{ABCC} = 15600\ ns + (t_{tout} * 734\ ns) + \sum_{n=1}^{ttout} (data\ length\ of\ n * 35\ ns)$

$t_{tout}$ = *total number of output submodules*

### Calculating Input Processing Time

This needs to be measured by the application designer. The time consists of two delays:

- **The delay added by the module**
  The time from when the message is available in the module until it is available on the network.
- **The delay added by the application**
  This is the time it takes for the application from inputs are captured until the inputs are available for the module.

To calculate the delay added by the module the following formula shall be used:

$t_{ABCC}$= 12980 ns + ($t_{tin}$ * 789 ns ) + $\sum_{n=1}^{ttin}(data\ length\ of\ n * 35\ ns)$ + ($t_{tout}$ * 240 ns )

$t_{tout}$ = *total number of output submodules*

$t_{tin}$ = *total number of input submodules*

### GSDML entries

When the total time has been calculated for every module, the following must be added to the GSDML file:

IsochroneModeInRT_Classes=”RT_CLASS_3” in the <InterfacesubmoduleItem>

<IsochroneMode T_DC_Base="8" T_DC_Min="1" T_DC_Max="16" T_IO_Base="1000" T_IO_OutputMin="x" T_IO_InputMin="y"/>

X = *output processing time*

Y = *input processing time*

### Using the Sync attributes

**Nonisochronous mode**

In this mode, the isochronous mode is not enabled in the PLC configuration tool. The application does not have to implement any of the attributes in the Sync object. The module will still write to attribute 1, Cycle time, but it will disregard the answer from the application.

**Isochronous mode**

In this mode the isochronous mode is enabled in the PLC configuration tool. The application must then implement all attributes of the Sync object. The first attribute read by the module will be #8, supported sync modes. The application must respond with "1" for the isochronous AR to be established.

By start-up: attribute #8, supported sync modes, is read:

1. **For every submodule where ISOM is activated in the PLC:**
   - Read OutputProcessing
   - Read InputProcessing
   - Read MinCycleTime
2. **Write sync mode**
   - NACK && ISOM enabled in PLC: NACK
3. **Write cycle time**
   - NACK && ISOM enabled in PLC: NACK
4. **If ISOM is activated:**
   - Set OutputValid
     - If NACK: NACK
   - Set InputCapture
     - If NACK: NACK

## Supported Commands

| | |
|---|---|
| Object: | Get_Attribute |
| Instance: | Get_Attribute<br>Set_Attribute |

## Object Attributes (Instance #0)

(Consult the general Anybus CompactCom 40 Software Design Guide for further information.)

## Instance Attributes (Instance #1)

| # | Name | Access | Profinet IRT | Type | Description |
|---|---|---|---|---|---|
| 1 | Cycle time | Get/Set | RR*SCF | UINT32 | The module supports cycle times as low as 250 µs |
| 2 | Output valid | Get/Set | T_IO_Output | UINT32 | Output valid point relative to SYNC events, in nanoseconds<br><br>Default value: 0 |
| 3 | Input capture | Get/Set | T_IO_Input | UINT32 | Input capture point relative to SYNC events, in nanoseconds<br><br>Default value: 0 |
| 4 | Output processing | Get | T_IO_OutputMin | UINT32 | Minimum required time, in nanoseconds, between RDPDI interrupt and “Output valid” |
| 5 | Input processing | Get | T_IO_InputMin | UINT32 | Maximum required time, in nanoseconds, from “Input capture” until write process data has been completely written to the Anybus CompactCom module |
| 6 | Min cycle time | Get | T_DC_Min | UINT32 | The module supports cycle times as low as 250 µs |
| 7 | Sync mode | Get/Set | - | UINT16 | This attribute is used to select synchronization mode. It enumerates the bits in attribute 8<br>0: Nonsynchronous operation. (Default value if nonsynchronous operation is supported)<br>1: Synchronous operation<br>2 - 65535: Reserved. Any attempt to set sync mode to an unsupported value shall generate an error response |
| 8 | Supported sync modes | Get | - | UINT16 | A list of the synchronization modes the application supports.<br>Each bit corresponds to a mode in attribute 7<br>Bit 0: 1 = Nonsynchronous mode supported<br>Bit 1: 1 = Synchronous mode supported<br>Bit 2 - 15: Reserved (0) |

# 12.3 PROFINET IO Object (F6h)

## Categories

Basic, extended, advanced

## Object Description

This object implements PROFINET IO related settings in the host application.

The implementation of this object is optional; the host application can support none, some, or all of the attributes specified below. The module will attempt to retrieve the values of these attributes during start-up; if an attribute is not implemented in the host application, simply respond with an error message (06h, "Invalid CmdExt[0]"). In such a case, the module will use its default value.

If the module attempts to retrieve a value of an attribute not listed below, respond with an error message (06h, "Invalid CmdExt[0]").

See also...

- "Network PROFINET IO Object (0Eh)" on page 85
- "Flowchart - Record Data Access" on page 152
- Anybus CompactCom 40 Software Design Guide, "Error Codes"

## Supported Commands

| | | |
|---|---|---|
| Object: | Get Attribute | |
| | Get_Record | (See "Command Details: Get_Record" on page 134) |
| | Set_Record | (See "Command Details: Set_Record" on page 136) |
| | AR_Check_Ind | (See "Command Details: AR_Check_Ind" on page 138) |
| | Cfg_Mismatch_Ind | (See "Command Details: Cfg_Mismatch_Ind" on page 139) |
| | Expected_Ident_Ind | (See "Command Details: Expected_Ident_Ind" on page 140) |
| | End_Of_Prm_Ind | (See "Command Details: End_Of_Prm_Ind" on page 142) |
| | AR_Abort_Ind | (See "Command Details: AR_Abort_Ind" on page 143) |
| Instance: | Get Attribute | |

## Object Attributes (Instance #0)

| # | Name | Access | Data Type | Value |
|---|---|---|---|---|
| 1 | Name | Get | Array of CHAR | 'PROFINET IO' |
| 2 | Revision | Get | UINT8 | 01h |
| 3 | Number of instances | Get | UINT16 | 0001h |
| 4 | Highest instance no. | Get | UINT16 | 0001h |

## Instance Attributes (Instance #1)

### Basic

| # | Name | Access | Type | Default Value[a] | Comment |
|---|---|---|---|---|---|
| 1 | Device ID | Get | UINT16 | 0010h | Identifies the device.<br>(Assigned by manufacturer) |
| 2 | Vendor ID<br>(I&M Manufacturer ID) | Get | UINT16 | 010Ch | Identifies the device manufacturer.<br>(Assigned by the PNO) |
| 3 | Station Type | Get | Array of CHAR | 'ABCC40-PIR' | Characterizes the device.<br>(Assigned by manufacturer); up to 25 characters. |
| 8 | I&M Order ID | Get | Array of CHAR | 'ABCC40-PIR' | I&M0 Parameter:<br>Order ID of device; up to 20 characters. |
| 9 | I&M Serial Number | Get | Array of CHAR | Assigned during manufacturing | I&M0 Parameter:<br>Serial number of device; up to 16 characters. |
| 19 | System Description | Get | Array of CHAR | 'HMS Industrial Networks Anybus CompactCom 40' | Up to 255 characters. |

a. If an attribute is not implemented, this value will be used instead.

### GSD Entries

The GSDML entries below must match the values of the corresponding attributes in the PROFINET IO object.

- Attributes #1 and #2 corresponds the following entry in the GSDML file:
  <DeviceIdentity VendorID="0x010C" DeviceID="0x0010">

- Attributes #3 corresponds the following entry in the GSDML file:
  DNS_CompatibleName="ABCC40-PIR"

- Attributes #8 corresponds the following entry in the GSDML file:
  <OrderNumber Value="ABCC40-PIR"/>

### Extended

| # | Name | Access | Type | Default Value[a] | Comment |
|---|---|---|---|---|---|
| 7 | Record Data Mode | Get | UINT8 | 00h | This setting affects how Record Data requests are treated, and constitutes a bit field as follows:<br>Bit 0:Index 0... 7FFFh:<br>0: Normal Mode<br>1: Transparent Mode<br>Bit 1:Index AFF0h... AFFFh:<br>Reserved (replaced with the command IM_Options) |
| 17 | Port 1 MAC address[b] | Get | Array of UINT8 | - | 6 byte physical address value; overrides the preprogrammed Mac address. |
| 18 | Port 2 MAC address[b] | Get | Array of UINT8 | - | 6 byte physical address value; overrides the preprogrammed Mac address. |

a. If an attribute is not implemented, this value will be used instead.
b. The module is preprogrammed with a valid Mac address. To use that address, do *not* implement this attribute.

### Advanced

| # | Name | Access | Type | Default Value[a] | Comment |
|---|---|---|---|---|---|
| 4 | MaxAr | Get | UINT32 | 0003h | Max. no.of simultaneous ARs.<br>(Range 1... 3) |
| 5 | (Reserved) | - | - | - | Reserved for future use. |
| 6 | (Reserved) | - | - | - | Reserved for future use. |
| 10 | I&M Hardware Revision | Get | UINT16 | Hardware Rev. | I&M0 Parameter:<br>Hardware revision of device; FFFFh indicates availability of profile specific information |
| 11 | I&M Software Revision | Get | Array of CHAR | Software Rev. | I&M0 Parameter:<br>Software revision of device.<br>Byte:Value:Meaning:<br>0: 'V' Official release<br>'R' Revision<br>'P' Prototype<br>'U' Under test<br>'T' Test device<br>1: 0... 255 Major Version<br>2: 0... 255 Minor Version<br>3: 0... 255 Internal Change<br>**Note:** Version V255.255.255 indicates availability of profile specific information. |
| 12 | I&M Revision Counter | Get | UINT16 | 0000h | I&M0 Parameter:<br>Revision counter of device; a changed value marks a change of the hardware or its parameters. |
| 13 | I&M Profile ID[b] | Get | UINT16 | F600h (Generic Device) | I&M0 Parameter:<br>If the application supports a specific profile, this can be specified here.<br>**Note**: this will not change the module's behavior, since it does not handle profiles. |

| # | Name | Access | Type | Default Value[a] | Comment |
|---|---|---|---|---|---|
| 14 | I&M Profile Specific Type[b] | Get | UINT16 | Generic Profile:<br>0004h<br>(Communication Module) | I&M0 Parameter:<br>If the application supports a specific profile, the profile specific types is specified here.<br>**Note**: this will not change the module's behavior, since it does not handle profiles. |
| 15 | (Reserved) | - | - | - | Reserved, not used. |
| 16 | (Reserved) | - | - | - | Reserved, not used. |
| 20 | (Reserved) | - | - | - | Reserved, not used. |
| 21 | (Reserved) | - | - | - | Reserved, not used. |

a. If an attribute is not implemented, this value will be used instead.

b. The host application has to implement the corresponding functionality; the Anybus module in itself does not alter its behavior based on these parameters.

# Command Details: Get_Record

## Category

Extended

## Details

Command Code: 10h

Valid for: Object Instance

## Description

The module issues this command in the following situations:

- Module receives a Record Data Read request towards an API other than 0 (zero).
- Module receives a Record Data Read request towards API 0, but the record in question is handled in Transparent Mode.
  **Note**: See attribute #7 in "Instance Attributes (Instance #1)" on page 131 for more information about Transparent Mode.

See "Flowchart - Record Data Access" on page 152 for more information.

It is optional to implement support for this command. If not implemented, the original network request will be rejected and an error is returned to the IO Controller/Supervisor.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | AR handle (low byte) | Handle for Application Relationship (AR) |
| CmdExt[1] | AR handle (high byte) | |
| MsgData[0] | API (low word, low byte) | Application Process Instance (API) |
| MsgData[1] | API (low word, high byte) | |
| MsgData[2] | API (high word, low byte) | |
| MsgData[3] | API (high word, high byte) | |
| MsgData[4] | Slot (low byte) | Slot number of request |
| MsgData[5] | Slot (high byte) | |
| MsgData[6] | Subslot (low byte) | Subslot number of request |
| MsgData[7] | Subslot (high byte) | |
| MsgData[8] | Index (low byte) | Index of request |
| MsgData[9] | Index (high byte) | |
| MsgData[10] | Length (low byte) | Range: 1 - 1524 |
| MsgData[11][a] | Length (high byte) | |

a. MsgData[11] is only available if the length value exceeds 255.

- **Response Details (Success)**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0... 1] | (reserved) | (set to zero) |
| MsgData[0... n] | Data (up to 1524 bytes) | Data to be returned in the Record Data Read response. |

- **Response Details (Error)**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0... 1] | (reserved) | (set to zero) |
| MsgData[0] | FFh | Object specific error |
| MsgData[1] | Error Code 1 | See "Details: Error Code 1" on page 144 |
| MsgData[2] | Error Code 2 | User specific error code |
| MsgData[3] | Additional Data 1 | API specific. Set to zero if no Additional Data 1 is defined. |
| MsgData[4] | Additional Data 2 | User specific. Set to zero if no Additional Data 2 is defined. |

See also...

- "Command Details: Set_Record" on page 136
- "Flowchart - Record Data Access" on page 152

## Command Details: Set_Record

### Category

Extended

### Details

Command Code: 11h

Valid for: Object Instance

### Description

The module issues this command in the following situations:

- Module receives a Record Data Write request towards an API other than 0 (zero).
- Module receives a Record Data Write request towards API 0, but the record in question is handled in Transparent Mode.

See "Flowchart - Record Data Access" on page 152 for more information.

It is optional to implement support for this command. If not implemented, the original network request will be rejected and an error is returned to the IO Controller/Supervisor.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | AR handle (low byte) | Handle for Application Relationship (AR) |
| CmdExt[1] | AR handle (high byte) | |
| MsgData[0] | API (low word, low byte) | Application Process Instance (API) |
| MsgData[1] | API (low word, high byte) | |
| MsgData[2] | API (high word, low byte) | |
| MsgData[3] | API (high word, high byte) | |
| MsgData[4] | Slot (low byte) | Slot number of request |
| MsgData[5] | Slot (high byte) | |
| MsgData[6] | Subslot (low byte) | Subslot number of request |
| MsgData[7] | Subslot (high byte) | |
| MsgData[8] | Index (low byte) | Index of request |
| MsgData[9] | Index (high byte) | |
| MsgData[10] | (reserved) | Reserved; set to zero. |
| MsgData[11...n] | Data (up to 1512 bytes) | Data from the Record Data Write request. |

- **Response Details (Success)**

  (no data)

- **Response Details (Error)**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0... 1] | (reserved) | (set to zero) |
| MsgData[0] | FFh | Object specific error |
| MsgData[1] | Error Code 1 | See "Details: Error Code 1" on page 144 |
| MsgData[2] | Error Code 2 | User specific error code |
| MsgData[3] | Additional Data 1 | API specific. Set to zero if no Additional Data 1 is defined. |
| MsgData[4] | Additional Data 2 | User specific. Set to zero if no Additional Data 2 is defined. |

See also...

- "Command Details: Get_Record" on page 134
- "Flowchart - Record Data Access" on page 152

# Command Details: AR_Check_Ind

## Category

Advanced

## Details

Command Code: 14h

Valid for: Object Instance

## Description

The module issues this command to inform the host application that an Application Relationship (AR) is to be established. It is optional to implement support for this command.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | AR handle (low byte) | Handle for the Application Relationship (AR) |
| CmdExt[1] | AR handle (high byte) | |
| MsgData[0] | IP address (low word, low byte) | IP address of the remote station (IO Controller/Supervisor) |
| MsgData[1] | IP address (low word, high byte) | |
| MsgData[2] | IP address (high word, low byte) | |
| MsgData[3] | IP address (high word, high byte) | |
| MsgData[4] | AR Type (low byte) | Indicates the type of AR as follows: |
| MsgData[5] | AR Type (high byte) | Value:Meaning:<br>1: IO_AR_SINGLE<br>3: IO_AR_CIR<br>4: IO_AR_REDUNDANT_CONTROLLER<br>5: IO_AR_REDUNDANT_DEVICE<br>6: SUPERVISOR_AR |
| MsgData[6] | AR Properties (low word, low byte) | Bit-field indicating the properties of the AR as follows: |
| MsgData[7] | AR Properties (low word, high byte) | Bit 0-2:State:<br>0: Backup<br>1: Primary<br>Bit 3:Supervisor take over allowed:<br>0: Not allowed<br>1: Allowed<br>Bit 4:Parameterization server:<br>0: EXTERNAL_PRM_SERVER<br>1: CM_INITIATOR<br>Bit 5-6:Data rate:<br>0: AT_LEAST_100 Mbps<br>1: 100 Mbps<br>2: 1 Gbps<br>3: 10 Gbps<br>Bit 8:Device Access:<br>0: AR_CONTEXT<br>1: DEVICE_CONTEXT<br>Bit 9-10:Companion AR:<br>0: SINGLE_AR<br>1: FIRST_AR<br>2: COMPANION_AR |
| MsgData[8] | AR Properties (high word, low byte) | |
| MsgData[9] | AR Properties (high word, high byte) | |
| MsgData[10] | Remote station name length | Length of remote station name, in bytes |
| MsgData[11...n] | Remote station name | Remote station name (IO Controller/Supervisor) |

- **Response Details**

  (No data)

## Command Details: Cfg_Mismatch_Ind

### Category

Advanced

### Details

Command Code: 15h

Valid for: Object Instance

### Description

The module issues this command to inform the host application that the configuration in the IO Controller (i.e. the Expected Identification) does not match the configuration defined by the host application (i.e. the Real Identification).

It is optional to implement support for this command.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | AR handle (low byte) | Handle for the Application Relationship (AR) |
| CmdExt[1] | AR handle (high byte) | |
| MsgData[0] | API (low word, low byte) | Application Process Instance (API) |
| MsgData[1] | API (low word, high byte) | |
| MsgData[2] | API (high word, low byte) | |
| MsgData[3] | API (high word, high byte) | |
| MsgData[4] | Slot (low byte) | Slot number of mismatch |
| MsgData[5] | Slot (high byte) | |
| MsgData[6] | Subslot (low byte) | Subslot number of mismatch |
| MsgData[7] | Subslot (high byte) | |
| MsgData[8] | Expected Module Identifier (low word, low byte) | Module identifier (as stated in the GSD file) derived from the IO Controller configuration |
| MsgData[9] | Expected Module Identifier (low word, high byte) | |
| MsgData[10] | Expected Module Identifier (high word, low byte) | |
| MsgData[11] | Expected Module Identifier (high word, high byte) | |
| MsgData[12] | Expected Submodule Identifier (low word, low byte) | Submodule identifier (as stated in the GSD file) derived from the IO Controller configuration |
| MsgData[13] | Expected Submodule Identifier (low word, high byte) | |
| MsgData[14] | Expected Submodule Identifier (high word, low byte) | |
| MsgData[15] | Expected Submodule Identifier (high word, high byte) | |

- **Response Details**

  (No data)

# Command Details: Expected_Ident_Ind

## Category

Advanced

## Details

Command Code: 1Bh

Valid for: Object Instance

## Description

The module issues this command to inform the host application of the Expected Identification (Module/Submodule List) that the IO Controller will use for the established AR.

Note that this information may be split in multiple segments, which means that this command will be issued multiple times by the module, each time containing different parts of the configuration.

Expected_Ident_Ind is similar to AR_Info_Ind but uses a different segmentation protocol that shall be used for the 40 series concept.

For very large configurations the Expected Identification can not fit into one message (max message data size = 1524 bytes). If this happens the message will be truncated.

It is optional to implement support for this command.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | Reserved | For segmented messages the CmdExt[1] byte has been reserved for segment bits. |
| CmdExt[1] | Cmd segment bits | |
| MsgData[0] | AR handle (low byte) | Handle for the Application Relationship. |
| MsgData[1] | AR handle (high byte) | |
| MsgData[2...n] | Data field | The first two bytes in the initial block of the Data field indicates the number of modules in the configuration. Each module is represented by a Module block, followed by a number of Submodule blocks (provided that the module in question contains submodules).<br>See "Data Format" on page 141 for coding of the data field. |

- **Response Details**

  (No data)

### Data Format

When all data has been received, the resulting data shall be interpreted as follows:

| Type | Name | | | | Description |
|---|---|---|---|---|---|
| UINT16 | iNbrApi | | | | Number of APIs in configuration. |
| UINT32 | | lApiNbr | | | Initial module block including API number and number of module blocks in the API. |
| UINT16 | | iNbrMod | | | |
| UINT16 | | | iSlotNbr | | Module block (8 bytes), see below. |
| UINT16 | | | iNbrSubMod | | |
| UINT32 | | | lModIdent | | |
| UINT16 | | | | iSubSlotNbr | Submodule block (10 bytes), see below. |
| UINT32 | | | | lSubModIdent | |
| UINT16 | | | | iInDataLength | |
| UINT16 | | | | iOutDataLength | |

The initial API block (iNbrApi) defines the number of APIs in the configuration.

Each API has an initial module block, that includes information on the API number (iApiNbr) and the number of modules (or slots) in the API.

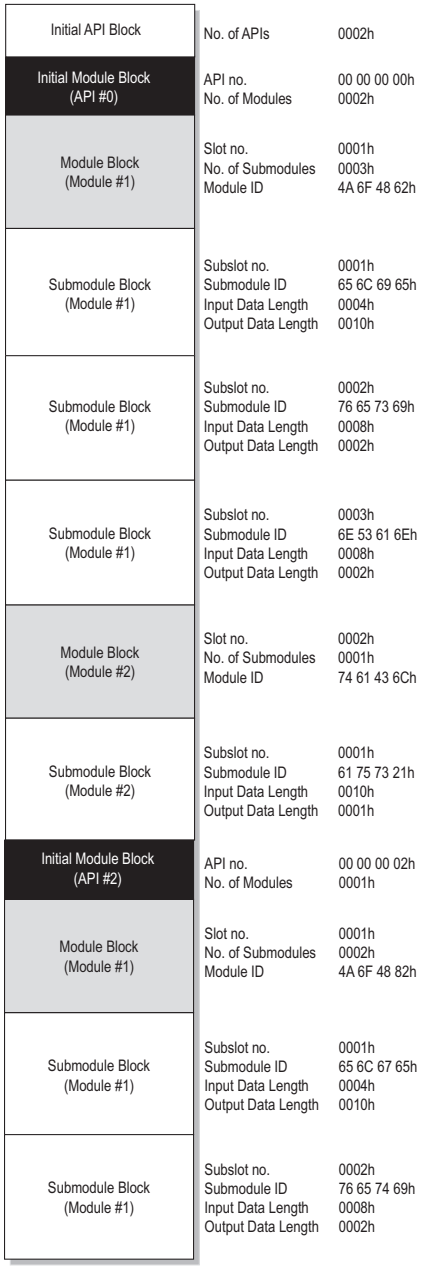
Each module starts with a module block, which holds the slot number, the number of submodules (or subslots) and the module identity number.

Finally each submodule block holds subslot number, submodule identification number, input and output data lengths.

*Example:*

In this example, the configuration contains two APIs with the following properties:

- API #1 contains two modules, the first with two submodules, the second with one submodule
- API #2 contains one module with one submodule



# Command Details: End_Of_Prm_Ind

## Category

Advanced

## Details

Command Code: 17h

Valid for: Object Instance

## Description

The module may issue this command to indicate to the host application that the parameterization phase is completed. It is optional to implement support for this command.

If implemented, the host application may, depending on the response issued to this command, be required to issue 'Appl_State_Ready' at a later stage to indicate that it is ready for data exchange. If not implemented, this is handled automatically by the module.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | AR handle (low byte) | Handle for the Application Relationship (AR) |
| CmdExt[1] | AR handle (high byte) | |
| MsgData[0] | API (low word, low byte) | Application Process Instance (API)<br><br>- Only valid if subslot > 0 |
| MsgData[1] | API (low word, high byte) | |
| MsgData[2] | API (high word, low byte) | |
| MsgData[3] | API (high word, high byte) | |
| MsgData[4] | Slot (low byte) | Slot number affected by the command<br>- Only valid if subslot > 0 |
| MsgData[5] | Slot (high byte) | |
| MsgData[6] | Subslot (low byte) | Subslot number affected by the command |
| MsgData[7] | Subslot (high byte) | Value:Meaning:<br>0: Command applies to all modules in the configuration<br>other: Command applies only to the specified slot/subslot |

- **Response Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | | |
| MsgData[0] | Application State | Value:Meaning:<br>0: Ready for Data Exchange<br>1: Not ready for data exchange<br>(Appl_State_Ready must be issued at a later stage) |

See also...

- "Command Details: Appl_State_Ready" on page 96

# Command Details: AR_Abort_Ind

## Category

Advanced

## Details

Command Code: 19h

Valid for: Object Instance

## Description

This command is optional. The module issues this command to indicate to the host application that an Application Relationship (AR) is aborted (by the application or any other source).

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | AR handle (low byte) | Handle for the Application Relationship (AR) |
| CmdExt[1] | AR handle (high byte) | |
| MsgData[0] | Reason code (low byte) | Reason code for the offline transition |
| MsgData[1] | Reason code (high byte) | Value:Reason:<br>0: No reason (unknown reason)<br>3: Out of mem<br>4: Add provider or consumer failed<br>5: Miss (consumer)<br>6: Cmi timeout<br>7: Alarm-open failed<br>8: Alarm-send.cnf(-)<br>9: Alarm-ack-send.cnf(-)<br>10: Alarm-data too long<br>11: Alarm.ind(err)<br>12: Rpc-client call.cnf(-)<br>13: Ar-abort.req<br>14: Re-run aborts existing<br>15: Got release.ind<br>16: Device passivated<br>17: Device/Ar removed<br>18: Protocol violation<br>19: NARE error<br>20: RPC-Bind error<br>21: RPC-Connect error<br>22: RPC-Read error<br>23: RPC-Write error<br>24: RPC-Control error<br>25: Forbidden pull or plug after check.rsp and before in-data.ind<br>26: AP removed<br>27: Link down<br>28: Could not register multicast-mac<br>29: Not synchronized (cannot start companion-ar)<br>30: Wrong topology (cannot start companion-ar)<br>31: Dcp, station-name changed<br>32: Dcp, reset to factory-settings<br>33: Cannot start companion AR because of parameter error |

- **Response Details**
  (no data)

## Details: Error Code 1

The error codes below shall be used when providing error responses to the following commands:

- Get_Record
- Set_Record
- Get_IM_Record
- Set_IM_Record

| High nibble (bits 4... 7) | | Low nibble (bits 0... 3) | | Comments |
|---|---|---|---|---|
| **ErrorClass** | **Meaning** | **ErrorCode** | **Meaning** | |
| 0... 9 | Reserved | (reserved) | (reserved) | - |
| 10 | Application | 0 | Read error | - |
| | | 1 | Write error | - |
| | | 2 | Module error | - |
| | | 3... 6 | (reserved) | - |
| | | 7 | Busy | - |
| | | 8 | Version conflict | - |
| | | 9 | Feature not supported | - |
| | | 10... 15 | User specific | - |
| 11 | Access | 0 | Invalid index | - |
| | | 1 | Write length error | - |
| | | 2 | Invalid slot/subslot | - |
| | | 3 | Type conflict | - |
| | | 4 | Invalid area | - |
| | | 5 | State conflict | - |
| | | 6 | Access denied | - |
| | | 7 | Invalid range | - |
| | | 8 | Invalid parameter | - |
| | | 9 | Invalid type | - |
| | | 10 | Backup | - |
| | | 11... 15 | User specific | - |
| 12 | Resource | 0 | Read constrain conflict | - |
| | | 1 | Write constrain conflict | - |
| | | 2 | Resource busy | - |
| | | 3 | Resource unavailable | - |
| | | 4... 7 | (reserved) | - |
| | | 8... 15 | User specific | - |
| 13... 15 | User specific | (user specific) | User specific | - |

See also...

- "Command Details: Get_Record" on page 134
- "Command Details: Set_Record" on page 136

# 12.4 Ethernet Host Object (F9h)

## Category

Extended, advanced

## Object Description

This object implements Ethernet features in the host application.

## Supported Commands

| | |
|---|---|
| Object: | Get_Attribute |
| Instance: | Get_Attribute<br>Set_Attribute |

## Object Attributes (Instance #0)

| # | Name | Access | Data Type | Value |
|---|---|---|---|---|
| 1 | Name | Get | Array of CHAR | 'Ethernet' |
| 2 | Revision | Get | UINT8 | 02h |
| 3 | Number of instances | Get | UINT16 | 0001h |
| 4 | Highest instance no. | Get | UINT16 | 0001h |

## Instance Attributes (Instance #1)

### Extended

| # | Name | Access | Type | Default[a] | Comment |
|---|---|---|---|---|---|
| 2 | Enable HICP | Get | BOOL | True | Enable/Disable HICP[b]<br>(see "HICP (Host IP Configuration Protocol)" on page 163). |
| 3 | Enable Web Server | Get | BOOL | True | Enable/Disable Web Server[b]<br>(see "Web Server" on page 33). |
| 5 | Enable Web ADI access | Get | BOOL | True | Enable/Disable Web ADI access[b]<br>(see "Web Server" on page 33). |
| 6 | Enable FTP server | Get | BOOL | True | Enable/Disable FTP server[b]<br>(see "FTP Server" on page 31). |
| 7 | Enable admin mode | Get | BOOL | False | Enable/Disable FTP admin mode[b]<br>(see "FTP Server" on page 31). |
| 8 | Network Status | Set | UINT16 | - | See "Network Status" on page 147. |

a. If an attribute is not implemented, the module will use this value instead
b. True=Enable/False=Disable

### Advanced

| # | Name | Access | Type | Default[a] | Comment |
|---|---|---|---|---|---|
| 1 | MAC address[b] | Get | Array of UINT8 | - | 6 byte physical address value; overrides the preprogrammed Mac address. Note that the new Mac address value must be obtained from the IEEE. |
| 9 | Port 1 MAC address[b] | Get | Array of UINT8 | - | 6 byte MAC address for port 1 (mandatory for the LLDP protocol).<br>Note: This setting overrides any Port MAC address in the host PROFINET IO Object. |
| 10 | Port 2 MAC address[b] | Get | Array of UINT8 | - | 6 byte MAC address for port 2 (mandatory for the LLDP protocol).<br>Note: This setting overrides any Port MAC address in the host PROFINET IO Object. |
| 11 | Enable ACD | Get | BOOL | 1 = True | Enable/Disable ACD protocol. |
| 12 | Port 1 State | Get | ENUM | 0 (Enabled) | The state of Ethernet port 1.<br>0: Enabled<br>1: Disabled |
| 13 | Port 2 State | Get | ENUM | 0 (Enabled) | The state of Ethernet port 2.<br>0: Enabled<br>1: Disabled |
| 14 | Enable firmware updated from web | Get | BOOL | 1 = True | Enables the firmware update HTTP page. |
| 15 | Enable reset from HICP | Get | BOOL | 0 = False | Enables the option to reset the module from HICP. |
| 16 | IP configuration | Set | Struct of<br>UINT32 IP address<br>UINT32 Subnet mask<br>UINT32 Gateway | N/A | Whenever the configuration is assigned or changed, the CompactCom module will update this attribute. |

a. If an attribute is not implemented, the module will use this value instead
b. The module is preprogrammed with a valid Mac address. To use that address, do *not* implement this attribute.

## Network Status

This attribute holds a bit field which indicates the overall network status as follows:

| Bit | Contents | Description |
|---|---|---|
| 0 | Link | Current global link status<br>1= Link sensed<br>0= No link |
| 1 | IP established | 1 = IP address established<br>0 = IP address not established |
| 2 | (reserved) | (mask off and ignore) |
| 3 | Link port 1 | Current link status for port 1<br>1 = Link sensed<br>0 = No link |
| 4 | Link port 2 | Current link status for port 2<br>1 = Link sensed<br>0 = No link |
| 5... 15 | (reserved) | (mask off and ignore) |

# A. Categorization of Functionality

The objects, including attributes and services, of the Anybus CompactCom and the application are divided into three categories: basic, advanced and extended.

## A.1 Basic

This category includes objects, attributes and services that are mandatory to implement or to use. They will be enough for starting up the Anybus CompactCom and sending/receiving data with the chosen network protocol. The basic functions of the industrial network are used.

Additional objects etc, that will make it possible to certify the product also belong to this category.

## A.2 Extended

Use of the objects in this category extends the functionality of the application. Access is given to the more specific characteristics of the industrial network, not only the basic moving of data to and from the network. Extra value is given to the application.

## A.3 Advanced

The objects, attributes and services that belong to this group offer specialized and/or seldom used functionality. Most of the available network functionality is enabled and accessible. Access to the specification of the industrial network is normally required.

# B. Anybus Implementation Details

## B.1 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. In the case of PROFINET IO, this bit is set when one or more IO connections are established.

## B.2 Anybus State Machine

The table below describes how the Anybus State Machine relates to the PROFINET IO network.

| Anybus State | Implementation | Comment |
|---|---|---|
| WAIT_PROCESS | The Anybus stays in this state until an IO connection with an IO controller is opened. | - |
| ERROR | Configuration data mismatch or initial parameter error. | - |
| PROCESS_ACTIVE | IO connection established with IO controller and valid output data has been provided at least once. | - |
| IDLE | IO controller with which an IO connection is established is in STOP mode or the IO controller has not provided valid output at least once. | Some IO controllers will not provide valid output data in the first cycles following a successful connection. |
| EXCEPTION | Turn module status LED red, to indicate major fault, turn network status LED off, and hold Ethernet MAC in reset. | Some kind of unexpected behavior, for example watchdog timeout. See also instance 1, attribute 7 in "Network Object (03h)" on page 76. |

## B.3 Application Watchdog Timeout Handling

Upon detection of an application watchdog timeout, the module will cease network participation and shift to state 'EXCEPTION'. No other network specific actions are performed.

# C. Message Segmentation

## C.1 General

**Category**: Advanced

The maximum message size supported by the Anybus CompactCom is 1500 bytes. To provide support for longer messages (needed when using the socket interface), a segmentation protocol is used.

The segmentation protocol is implemented in the message layer and must not be confused with the fragmentation used on the serial host interface. Consult the general Anybus CompactCom 40 Software Design Guide for further information.

The module supports 1 (one) simultaneous segmented message per instance.

## C.2 Command Segmentation

When a command message is segmented, the command initiator sends the same command header multiple times. For each message, the data field is exchanged with the next data segment.

Please note that some commands can't be used concurrently on the same instance, since they both need access to the segmentation buffer for that instance.

Command segmentation is used for the following commands:

- Send (see "Command Details: Send" on page 112)
- Send To (see "Command Details: Send_To" on page 113)

### Segmentation Control bits (Command)

| Bit | Contents | Meaning |
|---|---|---|
| 0 | FS | Set if the current segment is the first segment |
| 1 | LS | Set if the current segment is the last segment |
| 2 | AB | Set if the segmentation shall be aborted |
| 3...7 | (reserved) | Set to 0 (zero) |

### Segmentation Control bits (Response)

| Bit | Contents | Meaning |
|---|---|---|
| 0...7 | (reserved) | Ignore |

When issuing a segmented command, the following rules apply:

- When issuing the first segment, FS must be set.
- When issuing subsequent segments, both FS and LS must be cleared.
- When issuing the last segment, the LF-bit must be set.
- For single segment commands (i.e. size less or equal to 1500 bytes), both FS and LS must be set.
- The last response message contains the actual result of the operation.
- The command initiator may at any time abort the operation by issuing a message with AB set.

- If a segmentation error is detected during transmission, an error message is returned, and the current segmentation message is discarded. Note however that this only applies to the current segment; previously transmitted segments are still valid.

# C.3 Response Segmentation

When a response is segmented, the command initiator requests the next segment by sending the same command multiple times. For each response, the data field is exchanged with the next data segment.

Response segmentation is used for responses to the following commands:

- Receive (object specific, see "Command Details: Receive" on page 110)
- Receive From (object specific, see "Command Details: Receive_From" on page 111)

### Segmentation Control bits (Command)

| Bit | Contents | Meaning |
|---|---|---|
| 0 | (reserved) | (set to zero) |
| 1 | | |
| 2 | AB | Set if the segmentation shall be aborted |
| 3...7 | (reserved) | (set to zero) |

### Segmentation Control bits (Response)

| Bit | Contents | Meaning |
|---|---|---|
| 0 | FS | Set if the current segment is the first segment |
| 1 | LS | Set if the current segment is the last segment |
| 2...7 | (reserved) | (set to zero) |

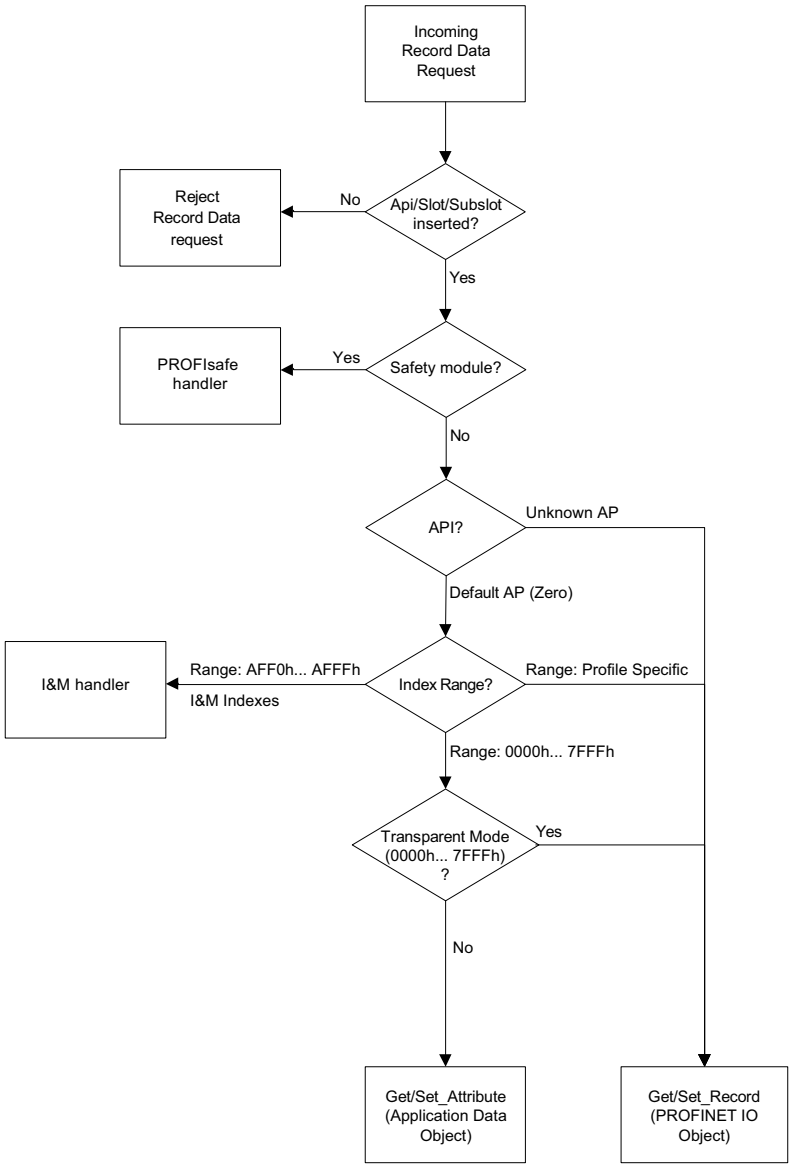
When receiving a segmented response, the following rules apply:

- In the first segment, FS is set.
- In all subsequent segment, both FS and LS are cleared.
- In the last segment, LS is set.
- For single segment responses (i.e. size less or equal to 1500 bytes), both FS and LS are set.
- The command initiator may at any time abort the operation by issuing a message with AB set.

# D. Flowcharts

## D.1 Flowchart - Record Data Access

This flowchart illustrates how Record Data requests are handled by the Anybus module.

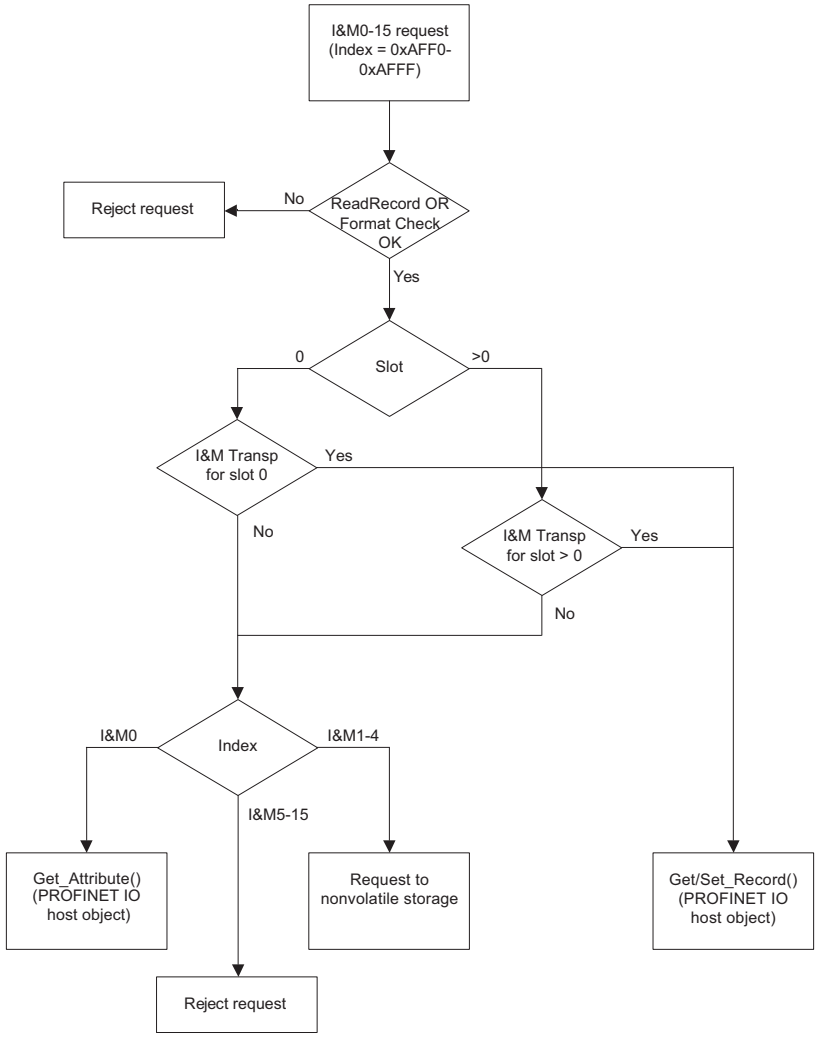



See also...


- "Application Data Instances (ADIs)" on page 16
- "PROFINET IO Object (F6h)" on page 130
- "Command Details: Get_Record" on page 134
- "Command Details: Set_Record" on page 136

# D.2 Flowchart - I&M Record Data Handling (API = 0)

This flowchart illustrates how I&M Record Data requests for API = 0 are handled by the Anybus module.




See also...

- "PROFINET IO Object (F6h)" on page 130
- "Command Details: Get_Record" on page 134
- "Command Details: Set_Record" on page 136

# E. Secure HICP (Secure Host IP Configuration Protocol)

## E.1 General

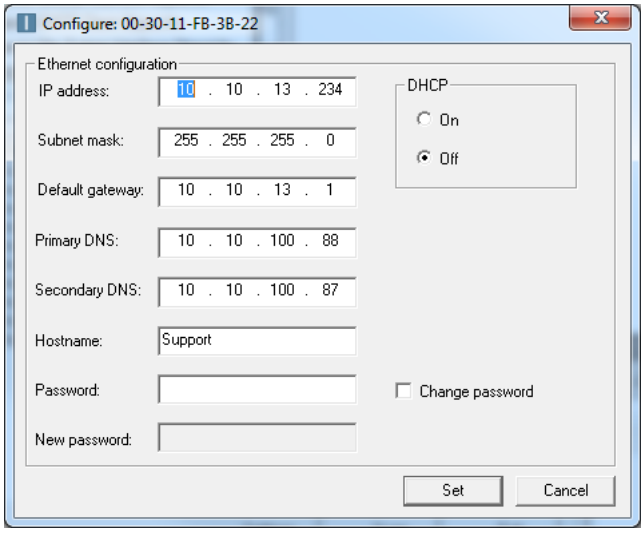
The module supports the Secure HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/disable DHCP. Anybus IPconfig can be downloaded free of charge from the HMS website, www.anybus.com. This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

The protocol offers secure authentication and the ability to restart/reboot the device(s).

## E.2 Operation

When the application is started, the network is automatically scanned for Anybus products. The network can be rescanned at any time by clicking "Scan".

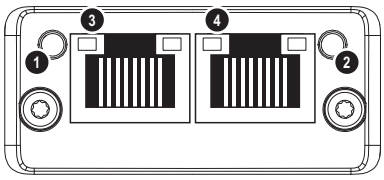
To alter the network settings of a module, double-click on its entry in the list. A window will appear, containing the settings for the module.




Validate the new settings by clicking "Set", or click "Cancel" to cancel all changes. Optionally, the configuration can be protected from unauthorized access by a password. To enter a password, check the "Change password" checkbox and enter the password in the "New password" text field.

# F. Technical Specification

## F.1 Front View (PROFINET IRT)

| # | Item | | Connector |
|---|---|---|---|
| 1 | Network Status LED[a] |  | Ethernet, RJ45 |
| 2 | Module Status LED[a] | | |
| 3 | Link/Activity LED (port 1) | | |
| 4 | Link/Activity LED (port 2) | | |

a. Test sequences are performed on the Network and Module Status LEDs during startup.

### F.1.1 Network Status LED

| LED State | Description | Comments |
|---|---|---|
| Off | Offline | - No power<br>- No connection with IO Controller |
| Green | Online (RUN) | - Connection with IO Controller established<br>- IO Controller in RUN state |
| Green, 1 flash | Online (STOP) | - Connection with IO Controller established<br>- IO Controller in STOP state or<br>- IO data bad<br>- IRT synchronization not finished |
| Green, blinking | Blink | Used by engineering tools to identify the node on the network |
| Red | Fatal event | Major internal error (this indication is combined with a red module status LED) |
| Red, 1 flash | Station Name error | Station Name not set |
| Red, 2 flashes | IP address error | IP address not set |
| Red, 3 flashes | Configuration error | Expected Identification differs from Real Identification |

### F.1.2 Module Status LED

| LED State | Description | Comments |
|---|---|---|
| Off | Not Initialized | No power - *or* - Module in 'SETUP' or 'NW_INIT' state |
| Green | Normal Operation | Module has shifted from the 'NW_INIT' state |
| Green, 1 flash | Diagnostic Event(s) | Diagnostic event(s) present |
| Red | Exception error | Device in state EXCEPTION |
| | Fatal event | Major internal error (this indication is combined with a red network status LED) |
| Alternating Red/Green | Firmware update | Do NOT power off the module. Turning the module off during this phase could cause permanent damage. |

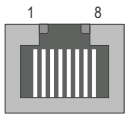
### F.1.3 LINK/Activity LED

| LED State | Description | Comments |
|---|---|---|
| Off | No Link | No link, no communication present |
| Green | Link | Ethernet link established, no communication present |
| Green, flickering | Activity | Ethernet link established, communication present |

### F.1.4 Ethernet Interface

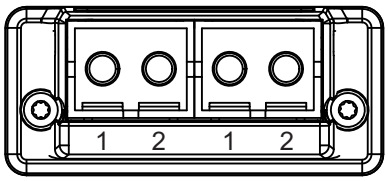
The Ethernet interface operates at 100 Mbit, full duplex, as required by PROFINET.

| Pin no | Description |
|---|---|
| 1, 2, 4, 5 | Connected to chassis ground over serial RC circuit |
| 3 | RD- |
| 6 | RD+ |
| 7 | TD- |
| 8 | TD+ |
| Housing | Cable Shield |




**IMPORTANT:**
*For information on how to connect the PROFINET cable, see "Front View (PROFINET IRT)" on page 155.*

## F.2 Front View (PROFINET IRT Fiber Optic)

| # | Item | | Connectors |
|---|---|---|---|
| 1 | Optical signal from the Anybus CompactCom module |   | SC-RJ Fiber Optic Connector |
| 2 | Optical signal to the Anybus CompactCom module | | |

# F.3 Protective Earth (PE) Requirements

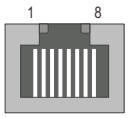
In order to ensure proper EMC behavior, the module must be properly connected to protective earth via the PE pad/PE mechanism described in the Anybus CompactCom 40 Hardware Design Guide. Proper EMC behavior is not guaranteed unless these PE requirements are fulfilled.

**Note:** The shield of the RJ45 connector is not connected directly to PE. As all nodes in a PROFINET network have to share chassis ground connection, the PROFINET cable shield has to be connected to the chassis ground at each node in the network.

For further information, see "PROFINET Installation Guideline for Cabling and Assembly", available for download at www.profinet.com.

### RJ45 Connector for PROFINET

| Pin no | Description |
|---|---|
| 1, 2, 4, 5 | Connected to chassis ground over serial RC circuit |
| 3 | RD- |
| 6 | RD+ |
| 7 | TD- |
| 8 | TD+ |
| Housing | Cable Shield |




# F.4 Power Supply

### Supply Voltage

The module requires a regulated 3.3 V power source as specified in the general Anybus CompactCom 40 Hardware Design Guide.

### Power Consumption

The PROFINET IO module is designed to fulfil the requirements of a Class B module. The current hardware design consumes up to 390 mA[1].

# F.5 Environmental Specification

Consult the Anybus CompactCom 40 Hardware Design Guide for further information.

# F.6 EMC Compliance

Consult the Anybus CompactCom 40 Hardware Design Guide for further information.

1. Note that in line with HMS policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. Note however that in any case, the Anybus CompactCom 40 PROFINET IO will remain as a Class B module.

# G. Conformance Test Guide

## G.1 General

When using the default settings of all parameters, the Anybus CompactCom 40 PROFINET IO module is precertified for network compliance. This precertification is done to ensure that your product *can* be certified.

Changes in the parameters in the example GSD file, supplied by HMS, will require a certification. A vendor ID can be obtained from PNO and is compulsory for certification. This chapter provides a guide for successful conformance testing your product, containing the Anybus CompactCom 40 PROFINET IO module, to comply with the demands for network certification set by the PNO.

Independent of selected operation mode, the actions described in this appendix have to be accounted for in the certification process. The identity of the product needs to be changed to match your company and device.

---

**IMPORTANT:** *This appendix provides guidelines and examples of what is needed for certification. Depending on the functionality of your application, there may be additional steps to take. Please contact HMS Industrial Networks at www.anybus.com for more information.*

# G.2 Reidentifying Your Product

After successful setting of the "Setup Complete" attribute in the Anybus Object (01h), the Anybus module asks for identification data from the host PROFINET IO Object (F6h). Therefore, the attributes listed below shall be implemented and proper values returned.

| Object/Instance | Attribute | Explanation | Default | Customer sample | Comment |
|---|---|---|---|---|---|
| PROFINET IO Object (F6h), Instance 1 | #1,Device ID | With this attribute you set the Device ID of the device | Device ID: 0010h | Device ID: YYYYh | This information must match the keys of the "DeviceIdentity" of the GSD-file. |
| PROFINET IO Object (F6h), Instance 1 | #2,Vendor ID | With this attribute you set the Vendor ID of the device | Vendor ID: 010Ch (HMS) | Vendor ID: XXXXh | Note that the GSD file keyword "VendorName" must correspond to the Vendor ID value. |
| PROFINET IO Object (F6h), Instance 1 | #3,Station Type | With this attribute you set the station type of the device | "ABCC40-PIR" | "Cust-PNIO-Dev" | This information matches, in the case of Anybus CompactCom 40 PROFINET IO, GSD keywords "DNS_CompatibleName" and "OrderNumber". The Station Type must be equal to the "DNS_CompatibleName", but it is allowed to have a completely different "OrderNumber", see also I&M Order ID below. |
| PROFINET IO Object (F6h), Instance 1 | #8, I&M Order ID | With this attribute you set the Order ID that is used in the I&M data | "ABCC40-PIR" | "Cust-PNIO-Dev " | This information must match the keys of the "OrderNumber" of the GSD-file. |
| PROFINET IO Object (F6h), Instance 1 | #10, I&M Hardware Revision | With this attribute you set the I&M Hardware Revision | (Hardware Rev.) | "0002h" | Optional. This information must match the keys of the "HardwareRelease" of the GSD-file. |
| PROFINET IO Object (F6h), Instance 1 | #11, I&M Software Revision | With this attribute you set the I&M Software Revision | (Software Rev.) | "V2.5.3" | Optional. This information must match the keys of the "SoftwareRelease" of the GSD-file. |
| PROFINET IO Object (F6h), Instance 1 | #19, System Description | With this attribute you set the description of the system | "HMS Industrial Networks Anybus CompactCom 40" | "Customer HMI Interface Module" | This information kan be read by means of SNMP from the network side. |

### Additional GSD File Information

The GSD file keyword "ProductFamily" shall correspond to the vendor's name of the device.

The GSD file keyword "MainFamily" lists the kinds of devices for which the product shall be listed. As of GSD specification v2.3, the following "families" are available:

"General", "Drives", "Switching Devices", "I/O", "Valves", "Controllers", "HMI", "Encoders", "NC/RC", "Gateway", "PLCs", "Ident Systems", "PA Profiles", "Network Components", "Sensors".

# G.3 Factory Default Reset

### Reset command to Application Object (FFh) must be supported

When PROFINET IO modules are delivered, they are required to be in their "Factory Default" state. For PROFINET devices this means that their Station Name is empty ("") and that the IP-suite is not assigned (IP 0.0.0.0). When a Factory Default Reset command is received from the network, the Anybus module will erase all IP and Station Name information and inform the host application that hardware or software reset of the Anybus module is required. This is done by sending a Reset command to the Application Object (FFh) of the host. For more details, please consult the Anybus CompactCom Software Design Guide.

## G.4 IP Address

Normally the IP numbers of PROFINET IO devices are assigned via the PROFINET network via DCP (Discovery and Configuration Protocol). HMS recommends not using the Network Configuration Object (04h, instances #3 - #6) during the initialization phase for PROFINET modules, unless the end user has requested the IP address to be set to a specific value (by for example using a keypad). The reason is that when a factory default reset command is received from the PROFINET network (via DCP) the node must be available after a hardware or software reset with the default IP-address (0.0.0.0).

## G.5 Station Name

Normally the Station Name of a PROFINET device is assigned by the end user via the PROFINET DCP protocol. HMS recommends not using the Station Name instance in the Network Configuration Object during the initialization phase for PROFINET modules. If this attribute is used, it is recommended that it is sent explicitly when the end user changes the Station Name with e.g. a keypad. The reason is that when a factory default reset command is received from the PROFINET network (via DCP), the node must be available after a hardware or software reset with the default Station Name ("").

---

**IMPORTANT**: *The Anybus module will forward all information about the connection being established to the IO Controller, as commands to the host PROFINET IO Object (F6h). Even though the host application might not need this information, a response must always be generated (such as 05h, "Unsupported command").*

## G.6 Documentation Considerations

To obtain a certificate, the following information must be present in the customer's user manual:

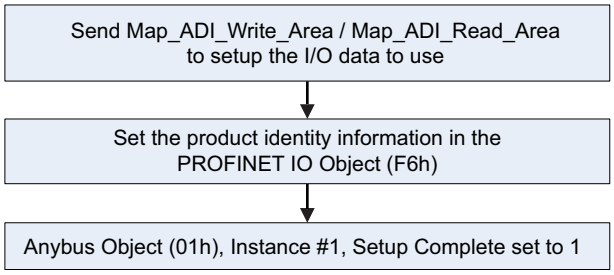
1. Behavior of the outputs if IOPS=BAD.
2. Behavior of the outputs if connection is aborted.
3. Behavior of the outputs at power on.

The Anybus CompactCom handles these situations in the following ways:

1. State change to IDLE. The network is informed that the I/O data of the submodule with IOPS=BAD is substituted with zeros (clear). No read process data is updated in the host interface.
2. State change to WAIT_PROCESS. The network is informed that the I/O data of all submodules is substituted with zeros (clear). No process data is updated in the host interface.
3. The network is informed that the I/O data of all submodules is substituted with zeros (clear). No process data is updated in the host interface.

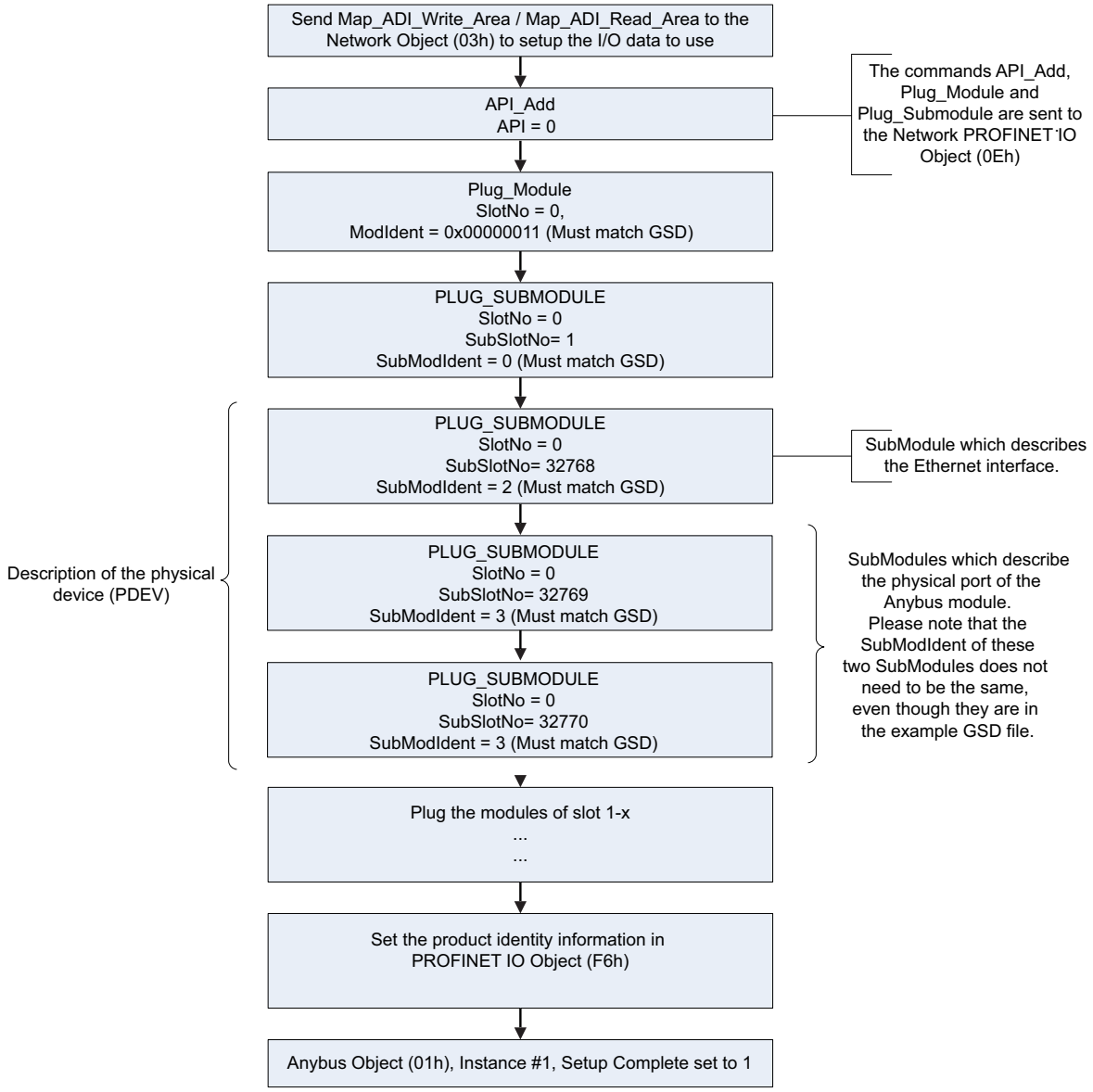
# G.7 Certification in Generic Anybus Mode

In Generic Anybus Mode (when the command API_add in the Network PROFINET IO Object (0Eh) is not used) there is normally nothing that needs to be considered apart from what is mentioned earlier in this appendix. The example HMS GSD file has to be modified with respect to the process mapping and identity of the product and this requires a certification of the product.



# G.8 Certification in Advanced Mode

In advanced mode (Network PROFINET IO Object (0Eh) is used), the most important thing is to use a Device Access Point (DAP) that conform to PROFINET IO Specification v2.0 or later (DAP2). From specification version 2.0 it is possible to describe the physical Ethernet interface and its ports (PDEV, or Physical Device) with a special mechanism. This is done with special submodules at slot 0 (the module at slot 0 is the access point for the device). HMS recommends following the flow below for setting up a DAP2.




The figure shows how to set up a PROFINET compatible DAP. Please note that for some commands only the relevant parameters are shown.

Please note that the values of "SubModIdent" in the above flowchart are the values of the example HMS GSD file. They can be changed if necessary, but there is no real need for it, the important thing is that it matches the GSD file. To be able to pass the PROFINET conformance test a "DAP2" is mandatory.

# H. Copyright Notices

This product includes software developed by Carnegie Mellon, the Massachusetts Institute of Technology, the University of California, and RSA Data Security:

libb (big.js)

The MIT Expat Licence.

Copyright (c) 2012 Michael Mclaughlin

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

**********

FatFs - FAT file system module  R0.09b  (C)ChaN, 2013

FatFs module is a generic FAT file system module for small embedded systems.
This is a free software that opened for education, research and commercial
developments under license policy of following trems.

Copyright (C) 2013, ChaN, all right reserved.

The FatFs module is a free software and there is NO WARRANTY.
No restriction on use. You can use, modify and redistribute it for
personal, non-profit or commercial products UNDER YOUR RESPONSIBILITY.
Redistributions of source code must retain the above copyright notice.

**********

**********

lwIP is licenced under the BSD licence:



**********

MD5 routines



L. Peter Deutsch
ghost@aladdin.com